

TMS320F2837xD Dual-Core Delfino Microcontrollers

Technical Reference Manual



Literature Number: SPRUHM8D
December 2013–Revised June 2015

Preface	77
1 C28x Processor	79
1.1 Overview	80
1.1.1 Floating-Point Unit	80
1.1.2 Trigonometric Math Unit	80
1.1.3 Viterbi, Complex Math, and CRC Unit II (VCU-II)	81
2 System Control	82
2.1 Introduction	83
2.2 System Control Functional Description	83
2.2.1 Device Identification	83
2.2.2 Device Configuration Registers	84
2.3 Resets	84
2.3.1 Reset Sources	84
2.3.2 External Reset (XRS)	85
2.3.3 Power-On Reset (POR)	85
2.3.4 Debugger Reset (SYSRS)	85
2.3.5 Watchdog Reset (WDRS)	85
2.3.6 NMI Watchdog Reset (NMIWDRS)	86
2.3.7 DCSM Safe Code Copy Reset (SCCRESET)	86
2.3.8 Hibernate Reset (HIBRESET)	86
2.3.9 Hardware BIST Reset (HWBISTR)	86
2.3.10 Test Reset (TRST)	86
2.4 Peripheral Interrupts	86
2.4.1 Interrupt Concepts	86
2.4.2 Interrupt Architecture	87
2.4.3 Interrupt Entry Sequence	88
2.4.4 Configuring and Using Interrupts	89
2.4.5 PIE Channel Mapping	91
2.4.6 Vector Tables	92
2.5 Exceptions and Non-Maskable Interrupts	98
2.5.1 Configuring and Using NMIs	98
2.5.2 Emulation Considerations	98
2.5.3 NMI Sources	99
2.5.4 Illegal Instruction Trap (ITRAP)	99
2.6 Safety Features	99
2.6.1 Write Protection on Registers	99
2.6.2 Missing Clock Detection Logic	100
2.6.3 PLLSLIP Detection	101
2.6.4 CPU1 and CPU2 PIE Vector Address Validity Check	101
2.6.5 NMIWDs	102
2.6.6 ECC and Parity Enabled RAMs, Shared RAMs Protection	102
2.6.7 ECC Enabled Flash Memory	102
2.6.8 Error Pin	102
2.7 Clocking	103
2.7.1 Clock Sources	104

2.7.2	Derived Clocks	106
2.7.3	Device Clock Domains	107
2.7.4	XCLKOUT.....	108
2.7.5	Clock Connectivity	108
2.7.6	Clock Source and PLL Setup	110
2.8	32-Bit CPU Timers 0/1/2.....	113
2.9	Watchdog Timers	114
2.9.1	Servicing the Watchdog Timer	114
2.9.2	Minimum Window Check	115
2.9.3	Watchdog Reset or Watchdog Interrupt Mode.....	115
2.9.4	Watchdog Operation in Low Power Modes	116
2.9.5	Emulation Considerations	116
2.10	Low Power Modes	117
2.10.1	IDLE	117
2.10.2	STANDBY	117
2.10.3	HALT	118
2.10.4	HIB.....	118
2.11	Memory Controller Module	119
2.11.1	Functional Description	120
2.12	Flash and OTP Memory	128
2.12.1	Features.....	128
2.12.2	Flash Tools	128
2.12.3	Default Flash Configuration	129
2.12.4	Flash Bank, OTP and Pump	129
2.12.5	Flash Module Controller (FMC)	129
2.12.6	Flash and OTP Automatic Power-Down Modes	130
2.12.7	Flash and OTP Performance.....	131
2.12.8	Flash Read Interface	132
2.12.9	Erase/Program Flash.....	134
2.12.10	Error Correction Code (ECC) Protection	136
2.12.11	Reserved Locations Within Flash and OTP	139
2.12.12	Procedure to Change the Flash Control Registers	139
2.13	Dual Code Security Module (DCSM).....	140
2.13.1	Functional Description	140
2.13.2	CSM Impact on Other On-Chip Resources	146
2.13.3	Incorporating Code Security in User Applications	147
2.14	Registers.....	152
2.14.1	Base Addresses	152
2.14.2	CPUTIMER_REGS Registers.....	153
2.14.3	PIE_CTRL_REGS Registers	160
2.14.4	WD_REGS Registers	187
2.14.5	NMI_INTERRUPT_REGS Registers.....	193
2.14.6	XINT_REGS Registers.....	205
2.14.7	DMA_CLA_SRC_SEL_REGS Registers	214
2.14.8	DEV_CFG_REGS Registers	221
2.14.9	CLK_CFG_REGS Registers	284
2.14.10	CPU_SYS_REGS Registers.....	306
2.14.11	ROM_PREFETCH_REGS Registers	340
2.14.12	DCSM_Z1_OTP Registers	342
2.14.13	DCSM_Z2_OTP Registers	349
2.14.14	DCSM_Z1_REGS Registers.....	356
2.14.15	DCSM_Z2_REGS Registers.....	376
2.14.16	DCSM_COMMON_REGS Registers	396

2.14.17	MEM_CFG_REGS Registers	403
2.14.18	ACCESS_PROTECTION_REGS Registers	447
2.14.19	MEMORY_ERROR_REGS Registers	466
2.14.20	ROM_WAIT_STATE_REGS Registers.....	483
2.14.21	FLASH_CTRL_REGS Registers	485
2.14.22	FLASH_ECC_REGS Registers	497
3	ROM Code and Peripheral Booting	519
3.1	Introduction	520
3.2	Device Boot Philosophy.....	520
3.3	Device Clocking on Power-up and Boot Time	520
3.4	Clock Configurations in Boot ROM	520
3.4.1	CPU1 Boot ROM Clocking Configuration.....	520
3.4.2	CPU2 Boot ROM Clocking Configuration	521
3.5	Faster Flash Power Up.....	521
3.6	Boot ROM DCSM init Sequence.....	521
3.6.1	CPU1 DCSM init Sequence	521
3.6.2	CPU2 DCSM init Sequence	521
3.7	Device Calibration on CPU1.....	522
3.8	CPU1 Boot ROM Procedure	522
3.9	CPU2 Boot ROM Procedure	523
3.10	Boot Modes Supported on CPU1	523
3.11	Boot Modes Supported on CPU2.....	528
3.12	CPU1 Boot ROM Flow Chart	530
3.13	CPU2 Boot ROM Flow Chart.....	533
3.14	Boot ROM Reset Causes and Handling	534
3.15	Exceptions and Interrupts handling.....	534
3.16	CPU1 OTP Boot Configure Word.....	535
3.17	Boot ROM Status information	535
3.17.1	Boot ROM Health and Status	535
3.17.2	CPU1 Boot ROM IPC NAK Status	536
3.17.3	CPU2 Boot ROM Health and Status	536
3.17.4	CPU2 Boot ROM IPC NAK status	537
3.18	Boot and IPC Commands	537
3.18.1	BOOTMODE Commands.....	537
3.18.2	CPU1 Boot ROM Supported IPC Commands	537
3.18.3	CPU2 Boot ROM Supported IPC Commands	539
3.19	Device Boot Process Timeline Diagram	541
3.20	Boot ROM GPIO Configurations:	542
3.21	Boot ROM Memory Map	543
3.21.1	F2837x Memory Map – CPU2	544
3.22	CPU1 and CPU2 ROM REVISION Information.....	544
3.23	RAM and Flash Usage and Application Entry Points	544
3.23.1	Reserved RAM Memory for CPU2-Boot ROM	545
3.23.2	CPU1 and CPU2 RAM Entry Point.....	545
3.23.3	CPU1 and CPU2 Flash Entry Point	545
3.23.4	CPU1/ and CPU2 Flash Reserved Memory.....	545
3.24	ROM Wait States	545
3.25	Device Boot Modes Description	546
3.25.1	Boot Data Stream Structure	546
3.25.2	Basic Data Transfer Procedure	550
3.25.3	CopyData Function	552
3.25.4	SCI Boot Mode.....	553
3.25.5	SPI Boot Mode	556

3.25.6	I2C Boot Mode	559
3.25.7	Parallel Boot Mode	562
3.25.8	CAN Boot Mode.....	566
3.25.9	USB Boot Mode.....	568
3.26	CLA Data ROM	569
3.26.1	CPU1/CPU2 CLA Data ROM	569
3.27	Secure ROM Contents	571
3.27.1	CPU1.Secure ROM.....	571
3.27.2	CPU2.Secure ROM.....	571
4	Direct Memory Access (DMA).....	572
4.1	Introduction	573
4.2	Architecture	574
4.2.1	Block Diagram.....	574
4.2.2	Common Peripheral Architecture	574
4.2.3	Peripheral Interrupt Event Trigger Sources	576
4.2.4	DMA Bus.....	582
4.3	Pipeline Timing and Throughput.....	582
4.4	CPU Arbitration	583
4.5	Channel Priority	584
4.5.1	Round-Robin Mode.....	584
4.5.2	Channel 1 High Priority Mode.....	584
4.6	Address Pointer and Transfer Control	585
4.7	Overrun Detection Feature	589
4.8	Register Descriptions.....	591
4.8.1	DMA Control Register (DMACTRL) — EALLOW Protected	592
4.8.2	Debug Control Register (DEBUGCTRL) — EALLOW Protected.....	593
4.8.3	Revision Register (REVISION).....	593
4.8.4	Priority Control Register 1 (PRIORITYCTRL1) — EALLOW Protected	594
4.8.5	Priority Status Register (PRIORITYSTAT)	595
4.8.6	Mode Register (MODE) — EALLOW Protected	596
4.8.7	Control Register (CONTROL) — EALLOW Protected	598
4.8.8	Burst Size Register (BURST_SIZE) — EALLOW Protected.....	600
4.8.9	BURST_COUNT Register	600
4.8.10	Source Burst Step Register Size (SRC_BURST_STEP) — EALLOW Protected	601
4.8.11	Destination Burst Step Register Size (DST_BURST_STEP) — EALLOW Protected	602
4.8.12	Transfer Size Register (TRANSFER_SIZE) — EALLOW Protected.....	602
4.8.13	Transfer Count Register (TRANSFER_COUNT)	603
4.8.14	Source Transfer Step Size Register (SRC_TRANSFER_STEP) — EALLOW Protected.....	603
4.8.15	Destination Transfer Step Size Register (DST_TRANSFER_STEP) — EALLOW Protected	604
4.8.16	Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) — EALLOW protected)	604
4.8.17	Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)	605
4.8.18	Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) — EALLOW Protected	605
4.8.19	Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) — All EALLOW Protected	606
4.8.20	Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR)	606
4.8.21	Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) — All EALLOW Protected	607
4.8.22	Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR).....	607
5	Control Law Accelerator (CLA).....	608
5.1	Control Law Accelerator (CLA) Overview	609
5.2	CLA Interface.....	611
5.2.1	CLA Memory	611

5.2.2	CLA Memory Bus	612
5.2.3	Shared Peripherals and EALLOW Protection	612
5.2.4	CLA Tasks and Interrupt Vectors	613
5.2.5	CLA Software Interrupt to CPU	615
5.3	CLA and CPU Arbitration	615
5.3.1	CLA and CPU Arbitration	615
5.3.2	CLA Message RAM	615
5.4	CLA Configuration and Debug	617
5.4.1	Building a CLA Application	617
5.4.2	Typical CLA Initialization Sequence	617
5.4.3	Debugging CLA Code	619
5.4.4	CLA Illegal Opcode Behavior	620
5.4.5	Resetting the CLA	620
5.5	Pipeline.....	621
5.5.1	Pipeline Overview.....	621
5.5.2	CLA Pipeline Alignment.....	621
5.5.3	Parallel Instructions.....	624
5.6	Instruction Set	626
5.6.1	Instruction Descriptions	626
5.6.2	Addressing Modes and Encoding.....	628
5.6.3	Instructions	630
5.7	Registers.....	741
5.7.1	CLA Base Addresses.....	741
5.7.2	CLA_REGS Registers.....	742
5.7.3	CLA_SOFTINT_REGS Registers.....	771
6	Inter-Processor Communication (IPC)	774
6.1	Inter-Processor Communication	775
6.2	Message RAMs	776
6.3	IPC Flags and Interrupts	776
6.4	IPC Command Registers	776
6.5	Flash Pump and Clock Configuration Semaphores	777
6.5.1	Flash Pump Semaphore	777
6.5.2	Clock Configuration Semaphore	777
6.5.3	Semaphore States	777
6.6	Free-Running Counter	778
6.7	IPC Communication Protocol.....	779
6.8	Registers.....	780
6.8.1	IPC Base Addresses	780
6.8.2	IPC_REGS_CPU1 Registers.....	781
6.8.3	IPC_REGS_CPU2 Registers.....	812
7	General-Purpose Input/Output (GPIO)	843
7.1	GPIO Overview	844
7.2	Configuration Overview	845
7.3	Digital General-Purpose I/O Control.....	845
7.4	Input Qualification.....	847
7.4.1	No Synchronization (Asynchronous Input)	847
7.4.2	Synchronization to SYSCLKOUT Only.....	847
7.4.3	Qualification Using a Sampling Window	847
7.5	USB Signals	850
7.6	SPI Signals	850
7.7	GPIO and Peripheral Muxing.....	851
7.8	Internal Pullup Configuration Requirements.....	852
7.9	Output X-BAR	854

7.9.1	Output X-BAR Architecture	855
7.10	Input X-BAR	856
7.11	Registers	858
7.11.1	GPIO Base Addresses	858
7.11.2	INPUT_XBAR_REGS Registers	859
7.11.3	OUTPUT_XBAR_REGS Registers	878
7.11.4	GPIO_CTRL_REGS Registers	970
7.11.5	GPIO_DATA_REGS Registers	1108
7.11.6	XBAR_REGS Registers	1157
8	Analog Subsystem	1178
8.1	Analog Subsystem	1179
8.1.1	Features	1179
8.1.2	Block Diagram	1179
8.1.3	Lock Registers	1182
8.2	Registers	1183
8.2.1	Analog Subsystem Base Addresses	1183
8.2.2	ANALOG_SUBSYS_REGS Registers	1184
9	Analog-to-Digital Converter (ADC)	1193
9.1	Analog-to-Digital Converter (ADC)	1194
9.1.1	Features	1194
9.1.2	ADC Block Diagram	1194
9.1.3	ADC Configurability	1195
9.1.4	SOC Principle of Operation	1199
9.1.5	SOC Configuration Examples	1202
9.1.6	ADC Conversion Priority	1204
9.1.7	Burst Mode	1207
9.1.8	EOC and Interrupt Operation	1209
9.1.9	Post-Processing Blocks	1209
9.1.10	Power-Up Sequence	1212
9.1.11	ADC Calibration	1212
9.2	ADC Timings	1213
9.2.1	ADC Timing Diagrams	1214
9.3	Additional Information	1218
9.3.1	Choosing an Acquisition Window Duration	1218
9.3.2	Achieving Simultaneous Sampling	1219
9.3.3	Designing an External Reference Circuit	1219
9.3.4	Internal Temperature Sensor	1220
9.4	Registers	1221
9.4.1	ADC Base Addresses	1221
9.4.2	ADC_REGS Registers	1222
9.4.3	ADC_RESULT_REGS Registers	1352
10	Buffered Digital to Analog Converter (DAC)	1373
10.1	Buffered Digital to Analog Converter (DAC) Overview	1374
10.1.1	Features	1374
10.1.2	Block Diagram	1374
10.2	Using the DAC	1374
10.3	Lock Registers	1375
10.4	Registers	1375
10.4.1	DAC Base Addresses	1375
10.4.2	DAC_REGS Registers	1376
11	Comparator Subsystem (CMPSS)	1384
11.1	CMPSS Overview	1385

11.1.1	Features	1385
11.1.2	Block Diagram	1385
11.2	Comparator.....	1386
11.3	Internal DAC	1386
11.4	Ramp Generator.....	1387
11.5	Digital Filter	1388
11.6	Registers	1390
11.6.1	CMPSS Base Addresses.....	1390
11.6.2	CMPSS_REGS Registers.....	1391
12	Sigma Delta Filter Module (SDFM)	1414
12.1	SDFM Module Overview	1415
12.1.1	SDFM Features.....	1415
12.1.2	Block Diagram	1416
12.2	Configuring Device Pins	1418
12.3	Input Control Unit.....	1419
12.3.1	Manchester Decoding	1421
12.4	Comparator Unit	1421
12.5	Data Filter Unit.....	1422
12.5.1	32-bit or 16-bit Data Filter Output Representation	1424
12.5.2	Data Rate and Latency of the Sinc Filter	1426
12.6	Interrupt Unit	1427
12.7	Register Descriptions	1429
12.8	Registers	1430
12.8.1	SDFM Base Addresses.....	1430
12.8.2	SDFM_REGS Registers.....	1431
13	Enhanced Pulse Width Modulator (ePWM)	1466
13.1	Introduction.....	1467
13.1.1	Submodule Overview	1468
13.2	Configuring Device Pins	1472
13.3	ePWM Submodules	1473
13.3.1	Overview	1473
13.3.2	Time-Base (TB) Submodule	1475
13.3.3	Counter-Compare (CC) Submodule	1486
13.3.4	Action-Qualifier (AQ) Submodule	1492
13.3.5	Dead-Band Generator (DB) Submodule	1507
13.3.6	PWM Chopper (PC) Submodule	1513
13.3.7	Trip-Zone (TZ) Submodule.....	1517
13.3.8	Event-Trigger (ET) Submodule	1522
13.3.9	Digital Compare (DC) Submodule	1528
13.3.10	EPWM X-BAR.....	1535
13.4	Applications to Power Topologies.....	1539
13.4.1	Overview of Multiple Modules	1539
13.4.2	Key Configuration Capabilities	1539
13.4.3	Controlling Multiple Buck Converters With Independent Frequencies	1540
13.4.4	Controlling Multiple Buck Converters With Same Frequencies	1542
13.4.5	Controlling Multiple Half H-Bridge (HHB) Converters	1544
13.4.6	Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)	1546
13.4.7	Practical Applications Using Phase Control Between PWM Modules.....	1549
13.4.8	Controlling a 3-Phase Interleaved DC/DC Converter	1550
13.4.9	Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter	1553
13.4.10	Controlling a Peak Current Mode Controlled Buck Module	1554
13.4.11	Controlling H-Bridge LLC Resonant Converter	1555
13.5	Registers	1557

13.5.1	EPWM Base Addresses	1557
13.5.2	EPWM_REGS Registers	1558
13.5.3	EPWM_XBAR_REGS Registers	1667
13.5.4	TRIG_REGS Registers	1751
14	High-Resolution Pulse Width Modulator (HRPWM)	1758
14.1	Introduction	1759
14.2	Operational Description of HRPWM	1761
14.2.1	Controlling the HRPWM Capabilities	1761
14.2.2	Configuring the HRPWM	1764
14.2.3	Configuring Hi-Res in Deadband Rising Edge and Falling Edge Delay	1765
14.2.4	Principle of Operation.....	1765
14.2.5	Deadband High Resolution Operation	1775
14.2.6	Scale Factor Optimizing Software (SFO)	1776
14.2.7	HRPWM Examples Using Optimized Assembly Code.	1776
14.3	Appendix A: SFO Library Software - SFO_TI_Build_V7.lib.....	1782
14.3.1	Scale Factor Optimizer Function - int SFO()	1782
14.3.2	Software Usage	1783
15	Enhanced Capture (eCAP).....	1785
15.1	Introduction.....	1786
15.2	Description	1786
15.3	Configuring Device Pins for the eCAP	1786
15.4	Capture and APWM Operating Mode	1787
15.5	Capture Mode Description	1788
15.5.1	Event Prescaler.....	1789
15.5.2	Edge Polarity Select and Qualifier	1790
15.5.3	Continuous/One-Shot Control	1790
15.5.4	32-Bit Counter and Phase Control	1791
15.5.5	CAP1-CAP4 Registers	1792
15.5.6	Using SWSYNC with the ECAP Module	1792
15.5.7	Interrupt Control	1793
15.5.8	Shadow Load and Lockout Control	1795
15.5.9	APWM Mode Operation	1795
15.6	Application of the ECAP Module	1796
15.6.1	Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger.....	1796
15.6.2	Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger.....	1799
15.6.3	Example 3 - Time Difference (Delta) Operation Rising Edge Trigger.....	1801
15.6.4	Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger.....	1803
15.7	Application of the APWM Mode	1805
15.7.1	Example 1 - Simple PWM Generation (Independent Channel/s)	1805
15.8	Registers	1807
15.8.1	eCAP Base Addresses	1807
15.8.2	ECAP_REGS Registers	1808
16	Enhanced QEP (eQEP).....	1823
16.1	Introduction.....	1824
16.2	Configuring Device Pins	1826
16.3	Description	1826
16.3.1	EQEP Inputs	1826
16.3.2	Functional Description	1827
16.3.3	eQEP Memory Map	1828
16.4	Quadrature Decoder Unit (QDU)	1829
16.4.1	Position Counter Input Modes	1829
16.4.2	eQEP Input Polarity Selection	1832
16.4.3	Position-Compare Sync Output	1832

16.5	Position Counter and Control Unit (PCCU)	1832
16.5.1	Position Counter Operating Modes	1832
16.5.2	Position Counter Latch	1835
16.5.3	Position Counter Initialization	1837
16.5.4	eQEP Position-compare Unit	1837
16.6	eQEP Edge Capture Unit	1839
16.7	eQEP Watchdog.....	1842
16.8	Unit Timer Base	1842
16.9	eQEP Interrupt Structure	1843
16.10	Registers	1843
16.10.1	eQEP Base Addresses	1843
16.10.2	EQEP_REGS Registers	1844
17	Serial Peripheral Interface (SPI)	1873
17.1	SPI Module Overview.....	1874
17.2	SPI Module Signal Summary	1875
17.3	Overview of SPI Module Registers	1876
17.4	Configuring Device Pins	1876
17.5	SPI Operation.....	1876
17.5.1	Introduction to Operation	1877
17.5.2	SPI Module Slave and Master Operation Modes	1877
17.5.3	Initialization Upon Reset	1880
17.5.4	Data Format.....	1880
17.5.5	Baud Rate and Clocking Schemes	1881
17.5.6	Data Transfer Example	1883
17.6	SPI DMA Transfers	1884
17.6.1	Transmitting Data Using SPI with DMA	1885
17.6.2	Receiving Data Using SPI with DMA	1885
17.7	SPI Interrupts	1886
17.7.1	SPI Interrupt Control Bits.....	1887
17.8	SPI FIFO Description	1887
17.9	SPI High-Speed Mode	1888
17.9.1	GPIOs Required for High-Speed Mode	1888
17.9.2	Configuring the SPI for High-Speed Mode.....	1889
17.10	SPI 3-Wire Mode Description	1889
17.11	SPI STEINV Bit in Digital Audio Transfers	1891
17.12	SPI Waveforms.....	1892
17.13	Registers	1898
17.13.1	SPI Base Addresses	1898
17.13.2	SPI_REGS Registers	1899
18	Serial Communications Interface (SCI)	1917
18.1	Enhanced SCI Module Overview.....	1918
18.2	Architecture	1919
18.3	SCI Module Signal Summary	1920
18.4	Configuring Device Pins	1920
18.5	Multiprocessor and Asynchronous Communication Modes.....	1920
18.6	SCI Programmable Data Format	1921
18.7	SCI Multiprocessor Communication	1921
18.7.1	Recognizing the Address Byte	1922
18.7.2	Controlling the SCI TX and RX Features	1922
18.7.3	Receipt Sequence.....	1922
18.8	Idle-Line Multiprocessor Mode.....	1922
18.8.1	Idle-Line Mode Steps	1923
18.8.2	Block Start Signal	1923

18.8.3	Wake-UP Temporary (WUT) Flag	1923
18.8.4	Receiver Operation	1924
18.9	Address-Bit Multiprocessor Mode	1924
18.9.1	Sending an Address	1924
18.10	SCI Communication Format	1925
18.10.1	Receiver Signals in Communication Modes.....	1926
18.10.2	Transmitter Signals in Communication Modes.....	1926
18.11	SCI Port Interrupts	1927
18.12	SCI Baud Rate Calculations	1927
18.13	SCI Enhanced Features.....	1928
18.13.1	SCI FIFO Description	1928
18.13.2	SCI Auto-Baud	1929
18.13.3	Autobaud-Detect Sequence	1930
18.14	Registers	1931
18.14.1	SCI Base Addresses	1931
18.14.2	SCI_REGS Registers	1932
19	Inter-Integrated Circuit Module (I2C).....	1951
19.1	Introduction to the I2C Module	1952
19.1.1	Features	1952
19.1.2	Features Not Supported.....	1952
19.1.3	Functional Overview	1953
19.1.4	Clock Generation.....	1954
19.1.5	I2C Clock Divider Registers (I2CCLKL and I2CCLKH)	1955
19.2	Configuring Device Pins	1956
19.3	I2C Module Operational Details	1956
19.3.1	Input and Output Voltage Levels	1956
19.3.2	Data Validity	1956
19.3.3	Operating Modes	1956
19.3.4	I2C Module START and STOP Conditions	1957
19.3.5	Serial Data Formats.....	1958
19.3.6	NACK Bit Generation	1960
19.3.7	Clock Synchronization	1961
19.3.8	Arbitration	1961
19.3.9	Digital Loopback Mode	1962
19.4	Interrupt Requests Generated by the I2C Module	1963
19.4.1	Basic I2C Interrupt Requests	1963
19.4.2	I2C FIFO Interrupts	1964
19.5	Resetting or Disabling the I2C Module	1965
19.6	Registers	1966
19.6.1	I2C Base Addresses	1966
19.6.2	I2C_REGS Registers	1967
20	Multichannel Buffered Serial Port (McBSP)	1989
20.1	Overview.....	1990
20.1.1	Features of the McBSPs	1990
20.1.2	McBSP Pins/Signals	1991
20.2	Configuring Device Pins	1992
20.3	McBSP Operation	1992
20.3.1	Data Transfer Process of McBSPs	1993
20.3.2	Companding (Compressing and Expanding) Data.....	1994
20.3.3	Clocking and Framing Data	1995
20.3.4	Frame Phases	1998
20.3.5	McBSP Reception.....	2000
20.3.6	McBSP Transmission	2001

20.3.7	Interrupts and DMA Events Generated by a McBSP	2002
20.4	McBSP Sample Rate Generator	2002
20.4.1	Block Diagram	2003
20.4.2	Frame Synchronization Generation in the Sample Rate Generator	2006
20.4.3	Synchronizing Sample Rate Generator Outputs to an External Clock	2006
20.4.4	Reset and Initialization Procedure for the Sample Rate Generator	2008
20.5	McBSP Exception/Error Conditions	2009
20.5.1	Types of Errors	2009
20.5.2	Overrun in the Receiver	2009
20.5.3	Unexpected Receive Frame-Synchronization Pulse	2011
20.5.4	Overwrite in the Transmitter	2013
20.5.5	Underflow in the Transmitter.....	2014
20.5.6	Unexpected Transmit Frame-Synchronization Pulse.....	2015
20.6	Multichannel Selection Modes.....	2017
20.6.1	Channels, Blocks, and Partitions.....	2017
20.6.2	Multichannel Selection.....	2018
20.6.3	Configuring a Frame for Multichannel Selection	2018
20.6.4	Using Two Partitions.....	2018
20.6.5	Using Eight Partitions	2020
20.6.6	Receive Multichannel Selection Mode.....	2021
20.6.7	Transmit Multichannel Selection Modes	2021
20.6.8	Using Interrupts Between Block Transfers.....	2023
20.7	SPI Operation Using the Clock Stop Mode	2024
20.7.1	SPI Protocol.....	2024
20.7.2	Clock Stop Mode	2025
20.7.3	Bits Used to Enable and Configure the Clock Stop Mode	2025
20.7.4	Clock Stop Mode Timing Diagrams.....	2026
20.7.5	Procedure for Configuring a McBSP for SPI Operation	2028
20.7.6	McBSP as the SPI Master	2028
20.7.7	McBSP as an SPI Slave.....	2030
20.8	Receiver Configuration	2031
20.8.1	Programming the McBSP Registers for the Desired Receiver Operation	2031
20.8.2	Resetting and Enabling the Receiver.....	2032
20.8.3	Set the Receiver Pins to Operate as McBSP Pins	2032
20.8.4	Enable/Disable the Digital Loopback Mode.....	2033
20.8.5	Enable/Disable the Clock Stop Mode.....	2033
20.8.6	Enable/Disable the Receive Multichannel Selection Mode	2034
20.8.7	Choose One or Two Phases for the Receive Frame	2034
20.8.8	Set the Receive Word Length(s)	2035
20.8.9	Set the Receive Frame Length	2035
20.8.10	Enable/Disable the Receive Frame-Synchronization Ignore Function.....	2036
20.8.11	Set the Receive Companding Mode	2037
20.8.12	Set the Receive Data Delay	2038
20.8.13	Set the Receive Sign-Extension and Justification Mode	2040
20.8.14	Set the Receive Interrupt Mode.....	2041
20.8.15	Set the Receive Frame-Synchronization Mode.....	2041
20.8.16	Set the Receive Frame-Synchronization Polarity.....	2043
20.8.17	Set the Receive Clock Mode	2045
20.8.18	Set the Receive Clock Polarity.....	2046
20.8.19	Set the SRG Clock Divide-Down Value.....	2048
20.8.20	Set the SRG Clock Synchronization Mode.....	2048
20.8.21	Set the SRG Clock Mode (Choose an Input Clock).....	2049
20.8.22	Set the SRG Input Clock Polarity.....	2050

20.9	Transmitter Configuration	2050
20.9.1	Programming the McBSP Registers for the Desired Transmitter Operation	2050
20.9.2	Resetting and Enabling the Transmitter.....	2051
20.9.3	Set the Transmitter Pins to Operate as McBSP Pins	2052
20.9.4	Enable/Disable the Digital Loopback Mode.....	2052
20.9.5	Enable/Disable the Clock Stop Mode.....	2052
20.9.6	Enable/Disable Transmit Multichannel Selection	2053
20.9.7	Choose One or Two Phases for the Transmit Frame	2055
20.9.8	Set the Transmit Word Length(s)	2055
20.9.9	Set the Transmit Frame Length	2056
20.9.10	Enable/Disable the Transmit Frame-Synchronization Ignore Function.....	2057
20.9.11	Set the Transmit Companding Mode.....	2058
20.9.12	Set the Transmit Data Delay	2059
20.9.13	Set the Transmit DXENA Mode.....	2061
20.9.14	Set the Transmit Interrupt Mode	2061
20.9.15	Set the Transmit Frame-Synchronization Mode	2062
20.9.16	Set the Transmit Frame-Synchronization Polarity	2063
20.9.17	Set the SRG Frame-Synchronization Period and Pulse Width	2064
20.9.18	Set the Transmit Clock Mode	2065
20.9.19	Set the Transmit Clock Polarity	2065
20.10	Emulation and Reset Considerations	2066
20.10.1	McBSP Emulation Mode	2067
20.10.2	Resetting and Initializing McBSPs.....	2067
20.11	Data Packing Examples.....	2069
20.11.1	Data Packing Using Frame Length and Word Length	2069
20.11.2	Data Packing Using Word Length and the Frame-Synchronization Ignore Function	2071
20.12	Interrupt Generation	2071
20.12.1	McBSP Receive Interrupt Generation.....	2072
20.12.2	McBSP Transmit Interrupt Generation	2072
20.12.3	Error Flags	2073
20.12.4	McBSP Interrupt Enable Register	2073
20.13	McBSP Modes.....	2073
20.14	McBSP Registers	2075
20.14.1	McBSP Base Addresses.....	2075
20.14.2	Data Receive Registers (DRR[1,2])	2076
20.14.3	Data Transmit Registers (DXR[1,2])	2076
20.14.4	Serial Port Control Registers (SPCR[1,2])	2077
20.14.5	Receive Control Registers (RCR[1, 2])	2082
20.14.6	Transmit Control Registers (XCR1 and XCR2)	2084
20.14.7	Sample Rate Generator Registers (SRGR1 and SRGR2)	2087
20.14.8	Multichannel Control Registers (MCR[1,2])	2089
20.14.9	Pin Control Register (PCR).....	2094
20.14.10	Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH)	2096
20.14.11	Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, XCERH)	2098
21	Controller Area Network (CAN)	2101
21.1	Overview.....	2102
21.1.1	Features	2102
21.1.2	Functional Description	2102
21.1.3	Block Diagram	2103
21.2	Configuring Device Pins	2104
21.3	Operating Modes	2104
21.3.1	Software Initialization	2104

21.3.2	CAN Message Transfer (Normal Operation)	2104
21.3.3	Test Modes	2105
21.4	Multiple Clock Source	2108
21.5	Interrupt Functionality	2108
21.5.1	Message Object Interrupts	2109
21.5.2	Status Change Interrupts	2109
21.5.3	Error Interrupts	2109
21.6	Global Power-down Mode	2109
21.6.1	Entering Global Power-down Mode	2109
21.6.2	Wakeup from Global Power-down Mode	2109
21.7	Local Power-down Mode	2109
21.7.1	Entering Local Power-down Mode	2110
21.7.2	Wakeup from Local Power-down Mode	2110
21.8	Parity Check Mechanism	2110
21.8.1	Behavior on Parity Error	2110
21.9	Debug Mode	2111
21.10	Module Initialization	2111
21.11	Configuration of Message Objects	2111
21.11.1	Configuration of a Transmit Object for Data Frames	2112
21.11.2	Configuration of a Transmit Object for Remote Frames	2112
21.11.3	Configuration of a Single Receive Object for Data Frames	2112
21.11.4	Configuration of a Single Receive Object for Remote Frames	2112
21.11.5	Configuration of a FIFO Buffer	2113
21.12	Message Handling	2113
21.12.1	Message Handler Overview	2113
21.12.2	Receive/Transmit Priority	2114
21.12.3	Transmission of Messages in Event Driven CAN Communication	2114
21.12.4	Updating a Transmit Object	2114
21.12.5	Changing a Transmit Object	2115
21.12.6	Acceptance Filtering of Received Messages	2115
21.12.7	Reception of Data Frames	2115
21.12.8	Reception of Remote Frames	2116
21.12.9	Reading Received Messages	2116
21.12.10	Requesting New Data for a Receive Object	2116
21.12.11	Storing Received Messages in FIFO Buffers	2116
21.12.12	Reading from a FIFO Buffer	2117
21.13	CAN Bit Timing	2118
21.13.1	Bit Time and Bit Rate	2119
21.13.2	Configuration of the CAN Bit Timing	2123
21.14	Message Interface Register Sets	2126
21.14.1	Message Interface Register Sets 1 and 2	2127
21.14.2	IF3 Register Set	2128
21.15	Message RAM	2128
21.15.1	Structure of Message Objects	2128
21.15.2	Addressing Message Objects in RAM	2130
21.15.3	Message RAM Representation in Debug Mode	2131
21.16	Registers	2133
21.16.1	CAN Base Addresses	2133
21.16.2	CAN_REGS Registers	2134
22	Universal Serial Bus (USB) Controller	2186
22.1	Introduction	2187
22.2	Features	2187
22.2.1	Block Diagram	2187

22.2.2	Signal Description	2187
22.2.3	VBus Recommendations	2188
22.3	Functional Description	2189
22.3.1	Operation as a Device	2189
22.3.2	Operation as a Host	2193
22.3.3	DMA Operation	2196
22.3.4	Address/Data Bus Bridge	2196
22.4	Initialization and Configuration	2198
22.4.1	Pin Configuration	2198
22.4.2	Endpoint Configuration	2198
22.5	Register Map	2199
22.6	Register Descriptions	2204
22.6.1	USB Device Functional Address Register (USBFADDR), offset 0x000	2204
22.6.2	USB Power Management Register (USBPOWER), offset 0x001	2205
22.6.3	USB Transmit Interrupt Status Register (USBTXIS), offset 0x002	2207
22.6.4	USB Receive Interrupt Status Register (USBRXIS), offset 0x004	2208
22.6.5	USB Transmit Interrupt Enable Register (USBTXIE), offset 0x006	2209
22.6.6	USB Receive Interrupt Enable Register (USBRXIE), offset 0x008	2210
22.6.7	USB General Interrupt Status Register (USBIS), offset 0x00A	2211
22.6.8	USB Interrupt Enable Register (USBIE), offset 0x00B	2213
22.6.9	USB Frame Value Register (USBFRAME), offset 0x00C	2215
22.6.10	USB Endpoint Index Register (USBEPIDX), offset 0x00E	2215
22.6.11	USB Test Mode Register (USBTEST), offset 0x00F	2216
22.6.12	USB FIFO Endpoint n Register (USBFIFO[0]-USBFIFO[3])	2218
22.6.13	USB Device Control Register (USBDEVCTL), offset 0x060	2219
22.6.14	USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ), offset 0x062	2221
22.6.15	USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ), offset 0x063	2222
22.6.16	USB Transmit FIFO Start Address Register (USBTXFIFOADD), offset 0x064	2223
22.6.17	USB Receive FIFO Start Address Register (USBRXFIFOADD), offset 0x066	2224
22.6.18	USB Connect Timing Register (USBCONTIM), offset 0x07A	2225
22.6.19	USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF), offset 0x07D	2226
22.6.20	USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF), offset 0x07E	2226
22.6.21	USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[0]-USBTXFUNCADDR[3])	2227
22.6.22	USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[0]-USBTXHUBADDR[3])	2228
22.6.23	USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[0]-USBTXHUBPORT[3])	2229
22.6.24	USB Receive Functional Address Endpoint n Registers (USBRXFUNCADDR[1]-USBRXFUNCADDR[3])	2230
22.6.25	USB Receive Hub Address Endpoint n Registers (USBRXHUBADDR[1]-USBRXHUBADDR[3])	2231
22.6.26	USB Receive Hub Port Endpoint n Registers (USBRXHUBPORT[1]-USBRXHUBPORT[3])	2232
22.6.27	USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[1]-USBTXMAXP[3])	2233
22.6.28	USB Control and Status Endpoint 0 Low Register (USBCSRL0), offset 0x102	2234
22.6.29	USB Control and Status Endpoint 0 High Register (USBCSRH0), offset 0x103	2236
22.6.30	USB Receive Byte Count Endpoint 0 Register (USBCOUNT0), offset 0x108	2237
22.6.31	USB Type Endpoint 0 Register (USBTTYPE0), offset 0x10A	2237
22.6.32	USB NAK Limit Register (USBNAKLMT), offset 0x10B	2238
22.6.33	USB Transmit Control and Status Endpoint n Low Register (USBTXCSSL[1]-USBTXCSSL[3])	2239
22.6.34	USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[1]-USBTXCSRH[3])	2242
22.6.35	USB Maximum Receive Data Endpoint n Registers (USBRXMAXP[1]-USBRXMAXP[3])	2244
22.6.36	USB Receive Control and Status Endpoint n Low Register (USBRXCSSL[1]-USBRXCSSL[3])	2245
22.6.37	USB Receive Control and Status Endpoint n High Register (USBRXCSRH[1]-USBRXCSRH[3])	2248

22.6.38	USB Receive Byte Count Endpoint n Registers (USBRXCOUNT[1]-USBRXCOUNT[3])	2250
22.6.39	USB Host Transmit Configure Type Endpoint n Register (USBTXTYPE[1]-USBTXTYPE[3])	2251
22.6.40	USB Host Transmit Interval Endpoint n Register (USBTXINTERVAL[1]-USBTXINTERVAL[3]) ...	2252
22.6.41	USB Host Configure Receive Type Endpoint n Register (USBRXTYPE[1]-USBRXTYPE[3])	2253
22.6.42	USB Host Receive Polling Interval Endpoint n Register (USBRXINTERVAL[1]- USBRXINTERVAL[3])	2254
22.6.43	USB Request Packet Count in Block Transfer Endpoint n Registers (USBQPKTCOUNT[1]- USBQPKTCOUNT[3])	2255
22.6.44	USB Receive Double Packet Buffer Disable Register (USBXDPKTBUFFDIS), offset 0x340	2256
22.6.45	USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFFDIS), offset 0x342	2257
22.6.46	USB External Power Control Register (USBEPCC), offset 0x400	2258
22.6.47	USB External Power Control Raw Interrupt Status Register (USBEPCCRIS), offset 0x404	2260
22.6.48	USB External Power Control Interrupt Mask Register (USBEPCCIM), offset 0x408	2261
22.6.49	USB External Power Control Interrupt Status and Clear Register (USBEPCCISC), offset 0x40C .	2262
22.6.50	USB Device RESUME Raw Interrupt Status Register (USBDRRIS), offset 0x410	2263
22.6.51	USB Device RESUME Raw Interrupt Mask Register (USBDRIM), offset 0x414	2264
22.6.52	USB Device RESUME Interrupt Status and Clear Register (USBDRISC), offset 0x418	2265
22.6.53	USB General-Purpose Control and Status Register (USBGPCS), offset 0x41C	2266
22.6.54	USB DMA Select Register (USBDMASEL), offset 0x450	2267
23	Universal Parallel Port (uPP)	2269
23.1	Introduction	2270
23.1.1	Features Supported	2270
23.2	Configuring Device Pins	2271
23.3	Functional Description	2271
23.3.1	Functional Block Diagram	2271
23.3.2	Data Flow	2271
23.3.3	Clock Generation and Control	2272
23.4	IO Interface and System Requirements	2274
23.4.1	Pin Multiplexing	2274
23.4.2	Internal DMA Controller Description	2274
23.4.3	Protocol Description	2276
23.4.4	Data Format	2279
23.4.5	Reset Considerations	2279
23.4.6	Interrupt Support	2280
23.4.7	Emulation Considerations	2281
23.4.8	Transmit and Receive FIFOs	2281
23.4.9	Transmit and Receive Data (MSG) RAM	2281
23.4.10	Initialization and Operation	2282
23.5	Registers	2284
23.5.1	UPP Base Addresses	2284
23.5.2	UPP_REGS Registers	2285
24	External Memory Interface (EMIF)	2318
24.1	Introduction	2319
24.1.1	Purpose of the Peripheral	2320
24.1.2	Features	2320
24.1.3	Functional Block Diagram	2320
24.2	Configuring Device Pins	2321
24.3	EMIF Module Architecture	2321
24.3.1	EMIF Clock Control	2321
24.3.2	EMIF Requests	2321
24.3.3	EMIF Signal Descriptions	2322
24.3.4	EMIF Signal Multiplexing Control	2323
24.3.5	SDRAM Controller and Interface	2323

24.3.6	Asynchronous Controller and Interface	2336
24.3.7	Data Bus Parking.....	2348
24.3.8	Reset and Initialization Considerations.....	2348
24.3.9	Interrupt Support	2348
24.3.10	DMA Event Support	2350
24.3.11	EMIF Signal Multiplexing	2350
24.3.12	Memory Map	2350
24.3.13	Priority and Arbitration	2351
24.3.14	System Considerations	2352
24.3.15	Power Management	2353
24.3.16	Emulation Considerations	2353
24.4	Example Configuration	2354
24.4.1	Hardware Interface.....	2354
24.4.2	Software Configuration	2354
24.5	Registers	2362
24.5.1	EMIF Base Addresses.....	2362
24.5.2	EMIF_REGS Registers	2363
24.5.3	EMIF1_CONFIG_REGS Registers	2380
24.5.4	EMIF2_CONFIG_REGS Registers	2385
Revision History		2389

List of Figures

2-1.	Device Interrupt Architecture	87
2-2.	Interrupt Propagation Path	88
2-3.	Missing Clock Detection Logic	101
2-4.	Error Pin Diagram	103
2-5.	Clocking System	104
2-6.	Single-ended 3.3V External Clock.....	105
2-7.	External Crystal	105
2-8.	External Resonator	106
2-9.	AUXCLKIN	106
2-10.	CPU-Timers	113
2-11.	CPU-Timer Interrupts Signals and Output Signal	113
2-12.	CPU Watchdog Module	114
2-13.	Memory Architecture	120
2-14.	Arbitration Scheme on Global Shared Memories.....	122
2-15.	Arbitration Scheme on Local Shared Memories	123
2-16.	FMC Interface with Core, Bank and Pump	130
2-17.	Flash Prefetch Mode	133
2-18.	ECC Logic Inputs and Outputs.....	136
2-19.	Storage of Zone-Select Bits in OTP	143
2-20.	Location of Zone-Select Block Based on Link-Pointer	144
2-21.	CSM Password Match Flow (PMF).....	148
2-22.	ECSL Password Match Flow (PMF).....	150
2-23.	TIM Register	154
2-24.	PRD Register	155
2-25.	TCR Register	156
2-26.	TPR Register	158
2-27.	TPRH Register	159
2-28.	PIECTRL Register	161
2-29.	PIEACK Register.....	162
2-30.	PIEIER1 Register	163
2-31.	PIEIFR1 Register	164
2-32.	PIEIER2 Register	165
2-33.	PIEIFR2 Register	166
2-34.	PIEIER3 Register	167
2-35.	PIEIFR3 Register	168
2-36.	PIEIER4 Register	169
2-37.	PIEIFR4 Register	170
2-38.	PIEIER5 Register	171
2-39.	PIEIFR5 Register	172
2-40.	PIEIER6 Register	173
2-41.	PIEIFR6 Register	174
2-42.	PIEIER7 Register	175
2-43.	PIEIFR7 Register	176
2-44.	PIEIER8 Register	177
2-45.	PIEIFR8 Register	178
2-46.	PIEIER9 Register	179
2-47.	PIEIFR9 Register	180

2-48.	PIEIER10 Register.....	181
2-49.	PIEIFR10 Register.....	182
2-50.	PIEIER11 Register.....	183
2-51.	PIEIFR11 Register.....	184
2-52.	PIEIER12 Register.....	185
2-53.	PIEIFR12 Register.....	186
2-54.	SCSR Register	188
2-55.	WDCNTR Register	189
2-56.	WDKEY Register.....	190
2-57.	WDCR Register	191
2-58.	WDWCR Register	192
2-59.	NMICFG Register	194
2-60.	NMIFLG Register	195
2-61.	NMIFLGCLR Register	197
2-62.	NMIFLGFRG Register.....	199
2-63.	NMIWDCNT Register	201
2-64.	NMIWDPRD Register	202
2-65.	NMISHDFLG Register.....	203
2-66.	XINT1CR Register.....	206
2-67.	XINT2CR Register.....	207
2-68.	XINT3CR Register.....	208
2-69.	XINT4CR Register	209
2-70.	XINT5CR Register.....	210
2-71.	XINT1CTR Register	211
2-72.	XINT2CTR Register	212
2-73.	XINT3CTR Register	213
2-74.	CLA1TASKSRCSELLOCK Register	215
2-75.	DMACHSRCSELLOCK Register	216
2-76.	CLA1TASKSRCSEL1 Register	217
2-77.	CLA1TASKSRCSEL2 Register	218
2-78.	DMACHSRCSEL1 Register	219
2-79.	DMACHSRCSEL2 Register	220
2-80.	DEVCFGLOCK1 Register	223
2-81.	PARTIDL Register	225
2-82.	PARTIDH Register.....	226
2-83.	REVID Register	227
2-84.	DC0 Register	228
2-85.	DC1 Register	229
2-86.	DC2 Register	230
2-87.	DC3 Register	231
2-88.	DC4 Register	233
2-89.	DC5 Register	234
2-90.	DC6 Register	235
2-91.	DC7 Register	236
2-92.	DC8 Register	237
2-93.	DC9 Register	238
2-94.	DC10 Register.....	239
2-95.	DC11 Register.....	240
2-96.	DC12 Register.....	241

2-97. DC13 Register.....	242
2-98. DC14 Register.....	243
2-99. DC15 Register.....	244
2-100. DC17 Register.....	245
2-101. DC18 Register.....	246
2-102. DC19 Register.....	247
2-103. DC20 Register.....	248
2-104. PERCNF1 Register.....	250
2-105. FUSEERR Register.....	251
2-106. SOFTPRES0 Register	252
2-107. SOFTPRES1 Register	253
2-108. SOFTPRES2 Register	254
2-109. SOFTPRES3 Register	256
2-110. SOFTPRES4 Register	257
2-111. SOFTPRES6 Register	258
2-112. SOFTPRES7 Register	259
2-113. SOFTPRES8 Register	260
2-114. SOFTPRES9 Register	261
2-115. SOFTPRES11 Register	262
2-116. SOFTPRES13 Register	263
2-117. SOFTPRES14 Register	264
2-118. SOFTPRES16 Register	265
2-119. CPUSEL0 Register	266
2-120. CPUSEL1 Register	268
2-121. CPUSEL2 Register	269
2-122. CPUSEL3 Register	270
2-123. CPUSEL4 Register	271
2-124. CPUSEL5 Register	272
2-125. CPUSEL6 Register	273
2-126. CPUSEL7 Register	274
2-127. CPUSEL8 Register	275
2-128. CPUSEL9 Register	276
2-129. CPUSEL11 Register.....	277
2-130. CPUSEL12 Register.....	279
2-131. CPUSEL14 Register.....	280
2-132. CPU2RESCTL Register.....	281
2-133. RSTSTAT Register	282
2-134. LPMSTAT Register	283
2-135. CLKSEM Register	285
2-136. CLKCFGLOCK1 Register.....	286
2-137. CLKSRCCTL1 Register	288
2-138. CLKSRCCTL2 Register	290
2-139. CLKSRCCTL3 Register	292
2-140. SYSPLLCTL1 Register.....	293
2-141. SYSPLLMULT Register	294
2-142. SYSPLLSTS Register	295
2-143. AUXPLLCTL1 Register	296
2-144. AUXPLLMULT Register.....	297
2-145. AUXPLLSTS Register.....	298

2-146. SYSCLKDIVSEL Register	299
2-147. AUXCLKDIVSEL Register	300
2-148. PERCLKDIVSEL Register	301
2-149. XCLKOUTDIVSEL Register	302
2-150. LOSPCP Register	303
2-151. MCDCCR Register	304
2-152. X1CNT Register.....	305
2-153. CPUSYSLOCK1 Register	307
2-154. HIBBOOTMODE Register	310
2-155. IORESTOREADDR Register	311
2-156. PIEVERRADDR Register	312
2-157. PCLKCR0 Register	313
2-158. PCLKCR1 Register	315
2-159. PCLKCR2 Register	316
2-160. PCLKCR3 Register	318
2-161. PCLKCR4 Register	319
2-162. PCLKCR6 Register	320
2-163. PCLKCR7 Register	321
2-164. PCLKCR8 Register	322
2-165. PCLKCR9 Register	323
2-166. PCLKCR10 Register	324
2-167. PCLKCR11 Register	325
2-168. PCLKCR12 Register	326
2-169. PCLKCR13 Register	327
2-170. PCLKCR14 Register	328
2-171. PCLKCR16 Register	329
2-172. SECMSEL Register.....	330
2-173. LPMCR Register	331
2-174. GPIOLPMSEL0 Register.....	333
2-175. GPIOLPMSEL1 Register.....	335
2-176. TMR2CLKCTL Register	337
2-177. RESC Register	338
2-178. ROMPREFETCH Register.....	341
2-179. Z1OTP_LINKPOINTER1 Register.....	343
2-180. Z1OTP_LINKPOINTER2 Register.....	344
2-181. Z1OTP_LINKPOINTER3 Register.....	345
2-182. Z1OTP_PSWDLOCK Register	346
2-183. Z1OTP_CRCLOCK Register	347
2-184. Z1OTP_BOOTCTRL Register.....	348
2-185. Z2OTP_LINKPOINTER1 Register.....	350
2-186. Z2OTP_LINKPOINTER2 Register.....	351
2-187. Z2OTP_LINKPOINTER3 Register.....	352
2-188. Z2OTP_PSWDLOCK Register	353
2-189. Z2OTP_CRCLOCK Register	354
2-190. Z2OTP_BOOTCTRL Register.....	355
2-191. Z1_LINKPOINTER Register	357
2-192. Z1_OTPSECCLOCK Register	358
2-193. Z1_BOOTCTRL Register	359
2-194. Z1_LINKPOINTERERR Register	360

2-195. Z1_CSMKEY0 Register	361
2-196. Z1_CSMKEY1 Register	362
2-197. Z1_CSMKEY2 Register	363
2-198. Z1_CSMKEY3 Register	364
2-199. Z1_CR Register	365
2-200. Z1_GRABSECTR Register	366
2-201. Z1_GRABRAMR Register	369
2-202. Z1_EXEONLYSECTR Register	371
2-203. Z1_EXEONLYRAMR Register	374
2-204. Z2_LINKPOINTER Register	377
2-205. Z2_OTPSECLOCK Register	378
2-206. Z2_BOOTCTRL Register	379
2-207. Z2_LINKPOINTERERR Register	380
2-208. Z2_CSMKEY0 Register	381
2-209. Z2_CSMKEY1 Register	382
2-210. Z2_CSMKEY2 Register	383
2-211. Z2_CSMKEY3 Register	384
2-212. Z2_CR Register	385
2-213. Z2_GRABSECTR Register	386
2-214. Z2_GRABRAMR Register	389
2-215. Z2_EXEONLYSECTR Register	391
2-216. Z2_EXEONLYRAMR Register	394
2-217. FLSEM Register	397
2-218. SECTSTAT Register	398
2-219. RAMSTAT Register	401
2-220. DxLOCK Register	404
2-221. DxCOMMIT Register	405
2-222. DxACCPROT0 Register	406
2-223. DxTEST Register	407
2-224. DxINIT Register	408
2-225. DxINITDONE Register	409
2-226. LSxLOCK Register	410
2-227. LSxCOMMIT Register	411
2-228. LSxMSEL Register	413
2-229. LSxCLAPGM Register	415
2-230. LSxACCPROT0 Register	416
2-231. LSxACCPROT1 Register	418
2-232. LSxTEST Register	419
2-233. LSxINIT Register	421
2-234. LSxINITDONE Register	422
2-235. GSxLOCK Register	423
2-236. GSxCOMMIT Register	425
2-237. GSxMSEL Register	427
2-238. GSxACCPROT0 Register	429
2-239. GSxACCPROT1 Register	431
2-240. GSxACCPROT2 Register	433
2-241. GSxACCPROT3 Register	435
2-242. GSxTEST Register	437
2-243. GSxINIT Register	440

2-244. GSxINITDONE Register	442
2-245. MSGxTEST Register	444
2-246. MSGxINIT Register	445
2-247. MSGxINITDONE Register	446
2-248. NMAVFLG Register	448
2-249. NMAVSET Register	449
2-250. NMAVCLR Register	450
2-251. NMAVINTEN Register.....	451
2-252. NMCPURDAVADDR Register	452
2-253. NMCPUWRAVADDR Register.....	453
2-254. NMCPUFAVADDR Register.....	454
2-255. NMDMAWRAVADDR Register	455
2-256. NMCLA1RDAVADDR Register	456
2-257. NMCLA1WRAVADDR Register.....	457
2-258. NMCLA1FAVADDR Register.....	458
2-259. MAVFLG Register	459
2-260. MAVSET Register	460
2-261. MAVCLR Register	461
2-262. MAVINTEN Register	462
2-263. MCPUFAVADDR Register.....	463
2-264. MCPUWRAVADDR Register.....	464
2-265. MDMAWRAVADDR Register	465
2-266. UCERRFLG Register.....	467
2-267. UCERRSET Register.....	468
2-268. UCERRCLR Register	469
2-269. UCCPUREADDR Register	470
2-270. UCDMAREADDR Register	471
2-271. UCCLA1READDR Register	472
2-272. CERRFLG Register.....	473
2-273. CERRSET Register.....	474
2-274. CERRCLR Register	475
2-275. CCPUREADDR Register	476
2-276. CERRCNT Register	477
2-277. CERRTHRES Register.....	478
2-278. CEINTFLG Register	479
2-279. CEINTCLR Register	480
2-280. CEINTSET Register	481
2-281. CEINTEN Register.....	482
2-282. ROMWAITSTATE Register.....	484
2-283. FRDCNTL Register	486
2-284. FSPRD Register	487
2-285. FBAC Register	488
2-286. FBFALLBACK Register	489
2-287. FBPRDY Register	490
2-288. FPAC1 Register	491
2-289. FPAC2 Register	492
2-290. FMAC Register.....	493
2-291. FMSTAT Register.....	494
2-292. FRD_INTF_CTRL Register	496

2-293. ECC_ENABLE Register.....	498
2-294. SINGLE_ERR_ADDR_LOW Register.....	499
2-295. SINGLE_ERR_ADDR_HIGH Register.....	500
2-296. UNC_ERR_ADDR_LOW Register.....	501
2-297. UNC_ERR_ADDR_HIGH Register.....	502
2-298. ERR_STATUS Register.....	503
2-299. ERR_POS Register.....	505
2-300. ERR_STATUS_CLR Register.....	506
2-301. ERR_CNT Register.....	507
2-302. ERR_THRESHOLD Register.....	508
2-303. ERR_INTFLG Register.....	509
2-304. ERR_INTCLR Register.....	510
2-305. FDATAH_TEST Register.....	511
2-306. FDATAL_TEST Register.....	512
2-307. FADDR_TEST Register.....	513
2-308. FECC_TEST Register.....	514
2-309. FECC_CTRL Register.....	515
2-310. FOUTH_TEST Register.....	516
2-311. FOUTL_TEST Register.....	517
2-312. FECC_STATUS Register.....	518
3-1. BOOTCTRL Selection (applies to both the CPUs).....	528
3-2. Flow Chart (part 1).....	530
3-3. CPU1 Part 2.....	531
3-4. CPU1 Part 3.....	532
3-5. CPU2 Flow Chart (part 1).....	533
3-6. Boot ROM Memory Map.....	543
3-7. F2837x Memory Map – CPU2.....	544
3-8. Bootloader Basic Transfer Procedure.....	551
3-9. Overview of CopyData Function.....	552
3-10. Overview of SCI Bootloader Operation.....	553
3-11. Overview of SCI Boot Function.....	554
3-12. Overview of SCI_GetWordData Function.....	555
3-13. SPI Loader.....	556
3-14. Data Transfer From EEPROM Flow.....	557
3-15. Overview of SPIA_GetWordData Function.....	558
3-16. EEPROM Device at Address 0x50.....	559
3-17. Overview of I2C Boot Function.....	560
3-18. Random Read.....	561
3-19. Sequential Read.....	561
3-20. Overview of Parallel GPIO Bootloader Operation.....	562
3-21. Parallel GPIO Bootloader Handshake Protocol.....	563
3-22. Parallel GPIO Mode Overview.....	563
3-23. Parallel GPIO Mode - Host Transfer Flow.....	564
3-24. 8-Bit Parallel GetWord Function.....	565
3-25. Overview of CAN-A Bootloader Operation.....	566
3-26. USB Boot Flow.....	568
4-1. DMA Block Diagram.....	574
4-2. Common Peripheral Architecture.....	575
4-3. DMA Trigger Architecture.....	577

4-4.	Peripheral Interrupt Trigger Input Diagram	578
4-5.	4-Stage Pipeline DMA Transfer	582
4-6.	4-Stage Pipeline With One Read Stall (McBSP as source)	582
4-7.	DMA State Diagram	588
4-8.	Overrun Detection Logic	590
4-9.	DMA Control Register (DMACTRL)	592
4-10.	Debug Control Register (DEBUGCTRL)	593
4-11.	Revision Register (REVISION).....	593
4-12.	Priority Control Register 1 (PRIORITYCTRL1)	594
4-13.	Priority Status Register (PRIORITYSTAT)	595
4-14.	Mode Register (MODE)	596
4-15.	Control Register (CONTROL)	598
4-16.	Burst Size Register (BURST_SIZE)	600
4-17.	Burst Size Register (BURST_COUNT)	600
4-18.	Source Burst Step Size Register (SRC_BURST_STEP)	601
4-19.	Destination Burst Step Register Size (DST_BURST_STEP)	602
4-20.	Transfer Size Register (TRANSFER_SIZE).....	602
4-21.	Transfer Count Register (TRANSFER_COUNT)	603
4-22.	Source Transfer Step Size Register (SRC_TRANSFER_STEP)	603
4-23.	Destination Transfer Step Size Register (DST_TRANSFER_STEP)	604
4-24.	Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE)	604
4-25.	Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)	605
4-26.	Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP)	605
4-27.	Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW).....	606
4-28.	Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR)	606
4-29.	Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW)	607
4-30.	Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR).....	607
5-1.	CLA Block Diagram.....	610
5-2.	MVECT1 Register	743
5-3.	MVECT2 Register	744
5-4.	MVECT3 Register	745
5-5.	MVECT4 Register	746
5-6.	MVECT5 Register	747
5-7.	MVECT6 Register	748
5-8.	MVECT7 Register	749
5-9.	MVECT8 Register	750
5-10.	MCTL Register	751
5-11.	MIFR Register	752
5-12.	MIOVF Register	754
5-13.	MIFRC Register	755
5-14.	MICLR Register	756
5-15.	MICLROVF Register	757
5-16.	MIER Register	758
5-17.	MIRUN Register.....	760
5-18.	_MPC Register	761
5-19.	_MAR0 Register	762
5-20.	_MAR1 Register	763

5-21.	_MSTF Register	764
5-22.	_MR0 Register	767
5-23.	_MR1 Register	768
5-24.	_MR2 Register	769
5-25.	_MR3 Register	770
5-26.	SOFTINTEN Register	772
5-27.	SOFTINTFRC Register	773
6-1.	IPC Module Architecture	775
6-2.	Messaging with IPC Flags and Interrupts	776
6-3.	Flash Pump Semaphore State transitions	778
6-4.	Clock Configuration Semaphore State Transitions	778
6-5.	IPACK Register	782
6-6.	IPCSTS Register	785
6-7.	IPCSET Register	789
6-8.	IPCCLR Register	792
6-9.	IPCFLG Register	796
6-10.	IPCCOUNTERL Register	799
6-11.	IPCCOUNTERH Register	800
6-12.	IPSENDCOM Register	801
6-13.	IPSENDADDR Register	802
6-14.	IPSENDATA Register	803
6-15.	IPCREMOTEREPLY Register	804
6-16.	IPCRECVCOM Register	805
6-17.	IPCRECVADDR Register	806
6-18.	IPCRECVDATA Register	807
6-19.	IPCLOCALREPLY Register	808
6-20.	IPCBOOTSTS Register	809
6-21.	IPCBOOTMODE Register	810
6-22.	PUMPREQUEST Register	811
6-23.	IPACK Register	813
6-24.	IPCSTS Register	816
6-25.	IPCSET Register	820
6-26.	IPCCLR Register	823
6-27.	IPCFLG Register	827
6-28.	IPCCOUNTERL Register	830
6-29.	IPCCOUNTERH Register	831
6-30.	IPCRECVCOM Register	832
6-31.	IPCRECVADDR Register	833
6-32.	IPCRECVDATA Register	834
6-33.	IPCLOCALREPLY Register	835
6-34.	IPSENDCOM Register	836
6-35.	IPSENDADDR Register	837
6-36.	IPSENDATA Register	838
6-37.	IPCREMOTEREPLY Register	839
6-38.	IPCBOOTSTS Register	840
6-39.	IPCBOOTMODE Register	841
6-40.	PUMPREQUEST Register	842
7-1.	GPIO Logic for a Single Pin	844
7-2.	Input Qualification Using a Sampling Window	847

7-3.	Input Qualifier Clock Cycles	850
7-4.	Output X- Bar	854
7-5.	Output X-Bar Architecture	855
7-6.	Input X-BAR	857
7-7.	INPUT1SELECT Register	860
7-8.	INPUT2SELECT Register	861
7-9.	INPUT3SELECT Register	862
7-10.	INPUT4SELECT Register	863
7-11.	INPUT5SELECT Register	864
7-12.	INPUT6SELECT Register	865
7-13.	INPUT7SELECT Register	866
7-14.	INPUT8SELECT Register	867
7-15.	INPUT9SELECT Register	868
7-16.	INPUT10SELECT Register	869
7-17.	INPUT11SELECT Register	870
7-18.	INPUT12SELECT Register	871
7-19.	INPUT13SELECT Register	872
7-20.	INPUT14SELECT Register	873
7-21.	INPUTSELECTLOCK Register	874
7-22.	OUTPUT1MUX0TO15CFG Register	879
7-23.	OUTPUT1MUX16TO31CFG Register	882
7-24.	OUTPUT2MUX0TO15CFG Register	885
7-25.	OUTPUT2MUX16TO31CFG Register	888
7-26.	OUTPUT3MUX0TO15CFG Register	891
7-27.	OUTPUT3MUX16TO31CFG Register	894
7-28.	OUTPUT4MUX0TO15CFG Register	897
7-29.	OUTPUT4MUX16TO31CFG Register	900
7-30.	OUTPUT5MUX0TO15CFG Register	903
7-31.	OUTPUT5MUX16TO31CFG Register	906
7-32.	OUTPUT6MUX0TO15CFG Register	909
7-33.	OUTPUT6MUX16TO31CFG Register	912
7-34.	OUTPUT7MUX0TO15CFG Register	915
7-35.	OUTPUT7MUX16TO31CFG Register	918
7-36.	OUTPUT8MUX0TO15CFG Register	921
7-37.	OUTPUT8MUX16TO31CFG Register	924
7-38.	OUTPUT1MUXENABLE Register	927
7-39.	OUTPUT2MUXENABLE Register	931
7-40.	OUTPUT3MUXENABLE Register	935
7-41.	OUTPUT4MUXENABLE Register	939
7-42.	OUTPUT5MUXENABLE Register	943
7-43.	OUTPUT6MUXENABLE Register	947
7-44.	OUTPUT7MUXENABLE Register	951
7-45.	OUTPUT8MUXENABLE Register	955
7-46.	OUTPUTLATCH Register	959
7-47.	OUTPUTLATCHCLR Register	961
7-48.	OUTPUTLATCHFRC Register	963
7-49.	OUTPUTLATCHENABLE Register	965
7-50.	OUTPUTINV Register	967
7-51.	OUTPUTLOCK Register	969

7-52.	GPACTRL Register	973
7-53.	GPAQSEL1 Register	974
7-54.	GPAQSEL2 Register	975
7-55.	GPAMUX1 Register	976
7-56.	GPAMUX2 Register	977
7-57.	GPADIR Register	978
7-58.	GPAPUD Register	980
7-59.	GPAINV Register	982
7-60.	GPAODR Register	984
7-61.	GPAGMUX1 Register	986
7-62.	GPAGMUX2 Register	987
7-63.	GPACSEL1 Register	988
7-64.	GPACSEL2 Register	989
7-65.	GPACSEL3 Register	990
7-66.	GPACSEL4 Register	991
7-67.	GPALOCK Register	992
7-68.	GPACR Register	994
7-69.	GPBCTRL Register	996
7-70.	GPBQSEL1 Register	997
7-71.	GPBQSEL2 Register	998
7-72.	GPBMUX1 Register	999
7-73.	GPBMUX2 Register	1000
7-74.	GPBDIR Register	1001
7-75.	GPBPUD Register	1003
7-76.	GPBINV Register	1005
7-77.	GPBODR Register	1007
7-78.	GPBAMSEL Register	1009
7-79.	GPBGMUX1 Register	1011
7-80.	GPBGMUX2 Register	1012
7-81.	GPBCSEL1 Register	1013
7-82.	GPBCSEL2 Register	1014
7-83.	GPBCSEL3 Register	1015
7-84.	GPBCSEL4 Register	1016
7-85.	GPBLOCK Register	1017
7-86.	GPBCR Register	1019
7-87.	GPCCTRL Register	1021
7-88.	GPCQSEL1 Register	1022
7-89.	GPCQSEL2 Register	1023
7-90.	GPCMUX1 Register	1024
7-91.	GPCMUX2 Register	1025
7-92.	GPCDIR Register	1026
7-93.	GPCPUD Register	1028
7-94.	GPCINV Register	1030
7-95.	GPCODR Register	1032
7-96.	GPCGMUX1 Register	1034
7-97.	GPCGMUX2 Register	1035
7-98.	GPCCSEL1 Register	1036
7-99.	GPCCSEL2 Register	1037
7-100.	GPCCSEL3 Register	1038

7-101. GPCCSEL4 Register.....	1039
7-102. GPCLOCK Register.....	1040
7-103. GPCCR Register	1042
7-104. GPDCTRL Register	1044
7-105. GPDQSEL1 Register	1045
7-106. GPDQSEL2 Register	1046
7-107. GPDMUX1 Register.....	1047
7-108. GPDMUX2 Register.....	1048
7-109. GPDDIR Register	1049
7-110. GPDPUUD Register	1051
7-111. GPDINV Register.....	1053
7-112. GPDODR Register	1055
7-113. GPDGMUX1 Register	1057
7-114. GPDGMUX2 Register	1058
7-115. GPDCSEL1 Register.....	1059
7-116. GPDCSEL2 Register.....	1060
7-117. GPDCSEL3 Register.....	1061
7-118. GPDCSEL4 Register.....	1062
7-119. GPDLOCK Register.....	1063
7-120. GPDCCR Register	1065
7-121. GPECTRL Register	1067
7-122. GPEQSEL1 Register.....	1068
7-123. GPEQSEL2 Register.....	1069
7-124. GPEMUX1 Register.....	1070
7-125. GPEMUX2 Register.....	1071
7-126. GPEDIR Register.....	1072
7-127. GPEPUD Register.....	1074
7-128. GPEINV Register.....	1076
7-129. GPEODR Register	1078
7-130. GPEGMUX1 Register.....	1080
7-131. GPEGMUX2 Register.....	1081
7-132. GPECSEL1 Register.....	1082
7-133. GPECSEL2 Register.....	1083
7-134. GPECSEL3 Register.....	1084
7-135. GPECSEL4 Register.....	1085
7-136. GPELOCK Register.....	1086
7-137. GPECCR Register	1088
7-138. GPFCTRL Register	1090
7-139. GPFQSEL1 Register.....	1091
7-140. GPFMUX1 Register.....	1092
7-141. GPFDIR Register.....	1093
7-142. GPFPUUD Register.....	1095
7-143. GPFINV Register	1097
7-144. GPFODR Register	1099
7-145. GPFGMUX1 Register.....	1101
7-146. GPFCSEL1 Register.....	1102
7-147. GPFCSEL2 Register.....	1103
7-148. GPFLOCK Register	1104
7-149. GPFCCR Register	1106

7-150. GPADAT Register	1109
7-151. GPASET Register	1111
7-152. GPACLEAR Register	1113
7-153. GPATOGGLE Register	1115
7-154. GPBDAT Register	1117
7-155. GPBSET Register	1119
7-156. GPBCLEAR Register	1121
7-157. GPBTOGGLE Register	1123
7-158. GPCDAT Register	1125
7-159. GPCSET Register	1127
7-160. GPCCLEAR Register	1129
7-161. GPCTOGGLE Register	1131
7-162. GPDDAT Register	1133
7-163. GPDSET Register	1135
7-164. GPD CLEAR Register	1137
7-165. GPDTOGGLE Register	1139
7-166. GPEDAT Register	1141
7-167. GPESET Register	1143
7-168. GPECLEAR Register	1145
7-169. GPETOGGLE Register	1147
7-170. GPFDAT Register	1149
7-171. GPFSET Register	1151
7-172. GPFCLEAR Register	1153
7-173. GPFTOGGLE Register	1155
7-174. XBARFLG1 Register	1158
7-175. XBARFLG2 Register	1163
7-176. XBARFLG3 Register	1167
7-177. XBARCLR1 Register	1171
7-178. XBARCLR2 Register	1174
7-179. XBARCLR3 Register	1176
8-1. Analog Subsystem Block Diagram (337-Ball ZWT)	1180
8-2. Analog Subsystem Block Diagram (176-Pin PTP)	1181
8-3. INTOSC1TRIM Register	1185
8-4. INTOSC2TRIM Register	1186
8-5. TSNSCTL Register	1187
8-6. LOCK Register	1188
8-7. ANAREFTRIMA Register	1189
8-8. ANAREFTRIMB Register	1190
8-9. ANAREFTRIMC Register	1191
8-10. ANAREFTRIMD Register	1192
9-1. ADC Module Block Diagram	1195
9-2. SOC Block Diagram	1199
9-3. Single-Ended Input Model	1200
9-4. Differential Input Model	1200
9-5. Round Robin Priority Example	1205
9-6. High Priority Example	1206
9-7. ADC Burst Priority	1208
9-8. ADC EOC Interrupts	1209
9-9. ADC PPB Block Diagram	1210

9-10.	ADC PPB Interrupt Event	1211
9-11.	ADC Timings for 12-bit Mode in Early Interrupt Mode	1214
9-12.	ADC Timings for 12-bit Mode in Late Interrupt Mode	1215
9-13.	ADC Timings for 16-bit Mode in Early Interrupt Mode	1216
9-14.	ADC Timings for 16-bit Mode in Late Interrupt Mode (SYSCLK Cycles)	1217
9-15.	Shared Reference System	1220
9-16.	ADCCTL1 Register	1224
9-17.	ADCCTL2 Register	1225
9-18.	ADCBURSTCTL Register	1226
9-19.	ADCINTFLG Register	1228
9-20.	ADCINTFLGCLR Register	1230
9-21.	ADCINTOVF Register	1232
9-22.	ADCINTOVFCLR Register	1233
9-23.	ADCINTSEL1N2 Register	1234
9-24.	ADCINTSEL3N4 Register	1236
9-25.	ADCSOCPRCTL Register	1238
9-26.	ADCINTSOCSEL1 Register	1241
9-27.	ADCINTSOCSEL2 Register	1243
9-28.	ADCSOCFLG1 Register	1245
9-29.	ADCSOCFRC1 Register	1249
9-30.	ADCSOCOVF1 Register	1254
9-31.	ADCSOCOVFCLR1 Register	1257
9-32.	ADCSOC0CTL Register	1260
9-33.	ADCSOC1CTL Register	1263
9-34.	ADCSOC2CTL Register	1266
9-35.	ADCSOC3CTL Register	1269
9-36.	ADCSOC4CTL Register	1272
9-37.	ADCSOC5CTL Register	1275
9-38.	ADCSOC6CTL Register	1278
9-39.	ADCSOC7CTL Register	1281
9-40.	ADCSOC8CTL Register	1284
9-41.	ADCSOC9CTL Register	1287
9-42.	ADCSOC10CTL Register	1290
9-43.	ADCSOC11CTL Register	1293
9-44.	ADCSOC12CTL Register	1296
9-45.	ADCSOC13CTL Register	1299
9-46.	ADCSOC14CTL Register	1302
9-47.	ADCSOC15CTL Register	1305
9-48.	ADCEVTSTAT Register	1308
9-49.	ADCEVTCLR Register	1309
9-50.	ADCEVTSEL Register	1310
9-51.	ADCEVTINTSEL Register	1312
9-52.	ADCCOUNTER Register	1314
9-53.	ADCREV Register	1315
9-54.	ADCOFFTRIM Register	1316
9-55.	ADCPPB1CONFIG Register	1317
9-56.	ADCPPB1STAMP Register	1319
9-57.	ADCPPB1OFFCAL Register	1320
9-58.	ADCPPB1OFFREF Register	1321

9-59.	ADCPPB1TRIPHI Register	1322
9-60.	ADCPPB1TRIPLO Register.....	1323
9-61.	ADCPPB2CONFIG Register	1324
9-62.	ADCPPB2STAMP Register	1326
9-63.	ADCPPB2OFFCAL Register.....	1327
9-64.	ADCPPB2OFFREF Register.....	1328
9-65.	ADCPPB2TRIPHI Register	1329
9-66.	ADCPPB2TRIPLO Register.....	1330
9-67.	ADCPPB3CONFIG Register	1331
9-68.	ADCPPB3STAMP Register	1333
9-69.	ADCPPB3OFFCAL Register.....	1334
9-70.	ADCPPB3OFFREF Register.....	1335
9-71.	ADCPPB3TRIPHI Register	1336
9-72.	ADCPPB3TRIPLO Register.....	1337
9-73.	ADCPPB4CONFIG Register	1338
9-74.	ADCPPB4STAMP Register	1340
9-75.	ADCPPB4OFFCAL Register.....	1341
9-76.	ADCPPB4OFFREF Register.....	1342
9-77.	ADCPPB4TRIPHI Register	1343
9-78.	ADCPPB4TRIPLO Register.....	1344
9-79.	ADCINLTRIM1 Register	1345
9-80.	ADCINLTRIM2 Register	1346
9-81.	ADCINLTRIM3 Register	1347
9-82.	ADCINLTRIM4 Register	1348
9-83.	ADCINLTRIM5 Register	1349
9-84.	ADCINLTRIM6 Register	1350
9-85.	ADCRESULT0 Register	1353
9-86.	ADCRESULT1 Register	1354
9-87.	ADCRESULT2 Register	1355
9-88.	ADCRESULT3 Register	1356
9-89.	ADCRESULT4 Register	1357
9-90.	ADCRESULT5 Register	1358
9-91.	ADCRESULT6 Register	1359
9-92.	ADCRESULT7 Register	1360
9-93.	ADCRESULT8 Register	1361
9-94.	ADCRESULT9 Register	1362
9-95.	ADCRESULT10 Register.....	1363
9-96.	ADCRESULT11 Register.....	1364
9-97.	ADCRESULT12 Register.....	1365
9-98.	ADCRESULT13 Register.....	1366
9-99.	ADCRESULT14 Register.....	1367
9-100.	ADCRESULT15 Register.....	1368
9-101.	ADCPPB1RESULT Register.....	1369
9-102.	ADCPPB2RESULT Register.....	1370
9-103.	ADCPPB3RESULT Register.....	1371
9-104.	ADCPPB4RESULT Register.....	1372
10-1.	DAC Module Block Diagram	1374
10-2.	DACREV Register.....	1377
10-3.	DACCTL Register	1378

10-4. DACVALA Register	1379
10-5. DACVALS Register	1380
10-6. DACOUTEN Register	1381
10-7. DACLOCK Register	1382
10-8. DACTRIM Register	1383
11-1. CMPSS Module Block Diagram	1385
11-2. Comparator Digital Output	1386
11-3. DAC Reference Select	1386
11-4. Output Voltage Calculation	1386
11-5. Ramp Generator	1387
11-6. Ramp Generator Behavior	1388
11-7. Digital Filter Behavior	1388
11-8. COMPCTL Register	1392
11-9. COMPHYSCTL Register	1394
11-10. COMPSTS Register	1395
11-11. COMPSTSCLR Register	1396
11-12. COMPDACCTL Register	1397
11-13. DACHVALS Register	1398
11-14. DACHVALA Register	1399
11-15. RAMPMAXREFA Register	1400
11-16. RAMPMAXREFS Register	1401
11-17. RAMPDECVALA Register	1402
11-18. RAMPDECVALS Register	1403
11-19. RAMPSTS Register	1404
11-20. DACLVALS Register	1405
11-21. DACLVALA Register	1406
11-22. RAMPDLYA Register	1407
11-23. RAMPDLYS Register	1408
11-24. CTRIPLFILCTL Register	1409
11-25. CTRIPLFILCLKCTL Register	1410
11-26. CTRIPHFILCTL Register	1411
11-27. CTRIPHFILCLKCTL Register	1412
11-28. COMPLOCK Register	1413
12-1. Sigma Delta Filter Module (SDFM) CPU Interface	1415
12-2. Sigma Delta Filter Module (SDFM) Block Diagram	1417
12-3. Block Diagram of One Filter Module	1418
12-4. Typical PWM Interface to Sigma Delta Filter Module	1418
12-5. Manchester Coding Scheme	1421
12-6. Comparator Filter Resolution	1422
12-7. Frequency Response of Various Sinc Filters	1423
12-8. Data Filter Resolution	1424
12-9. SDFM Interrupt Unit	1428
12-10. SDIFLG Register	1432
12-11. SDIFLGCLR Register	1434
12-12. SDCTL Register	1435
12-13. SDMFILEN Register	1436
12-14. SDSTATUS Register	1437
12-15. SDCTLPARM1 Register	1438
12-16. SDDFPARM1 Register	1439

12-17. SDIPARM1 Register	1440
12-18. SDCMPH1 Register.....	1441
12-19. SDCMPL1 Register	1442
12-20. SDCPARM1 Register	1443
12-21. SDDATA1 Register	1444
12-22. SDCTLARM2 Register	1445
12-23. SDDFPARM2 Register	1446
12-24. SDIPARM2 Register	1447
12-25. SDCMPH2 Register.....	1448
12-26. SDCMPL2 Register	1449
12-27. SDCPARM2 Register	1450
12-28. SDDATA2 Register	1451
12-29. SDCTLARM3 Register	1452
12-30. SDDFPARM3 Register	1453
12-31. SDIPARM3 Register	1454
12-32. SDCMPH3 Register.....	1455
12-33. SDCMPL3 Register	1456
12-34. SDCPARM3 Register	1457
12-35. SDDATA3 Register	1458
12-36. SDCTLARM4 Register.....	1459
12-37. SDDFPARM4 Register	1460
12-38. SDIPARM4 Register	1461
12-39. SDCMPH4 Register.....	1462
12-40. SDCMPL4 Register	1463
12-41. SDCPARM4 Register	1464
12-42. SDDATA4 Register	1465
13-1. Multiple ePWM Modules.....	1469
13-2. Submodules and Signal Connections for an ePWM Module	1470
13-3. ePWM Submodules and Critical Internal Signal Interconnects	1472
13-4. Time-Base Submodule	1475
13-5. Time-Base Submodule Signals and Registers	1476
13-6. Time-Base Frequency and Period	1478
13-7. Time-Base Counter Synchronization Scheme 4	1480
13-8. Time-Base Up-Count Mode Waveforms	1482
13-9. Time-Base Down-Count Mode Waveforms	1483
13-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event...	1483
13-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event	1484
13-12. Global Reload: Signals and Registers.....	1485
13-13. Counter-Compare Submodule.....	1486
13-14. Detailed View of the Counter-Compare Submodule.....	1487
13-15. Counter-Compare Event Waveforms in Up-Count Mode	1490
13-16. Counter-Compare Events in Down-Count Mode	1491
13-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event	1492
13-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event	1492
13-19. Action-Qualifier Submodule	1493
13-20. Action-Qualifier Submodule Inputs and Outputs	1494
13-21. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs	1495

13-22. AQCTLR[SHDWAQAMODE]	1498
13-23. AQCTLR[SHDWAQBMODE]	1498
13-24. Up-Down-Count Mode Symmetrical Waveform	1500
13-25. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High	1501
13-26. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low	1502
13-27. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA	1503
13-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low	1504
13-29. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary	1505
13-30. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low	1506
13-31. Up-Down-Count, PWM Waveform Generation Utilizing T1 and T2 Events	1506
13-32. Dead_Band Submodule	1507
13-33. Configuration Options for the Dead-Band Submodule	1509
13-34. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)	1511
13-35. PWM Chopper Submodule	1513
13-36. PWM Chopper Submodule Operational Details	1514
13-37. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only	1514
13-38. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses	1515
13-39. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses	1516
13-40. Trip-Zone Submodule	1517
13-41. Trip-Zone Submodule Mode Control Logic	1521
13-42. Trip-Zone Submodule Interrupt Logic	1522
13-43. Event-Trigger Submodule	1523
13-44. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs	1524
13-45. Event-Trigger Interrupt Generator	1526
13-46. Event-Trigger SOCA Pulse Generator	1527
13-47. Event-Trigger SOCB Pulse Generator	1527
13-48. Digital-Compare Submodule High-Level Block Diagram	1528
13-49. GPIO MUX-to-Trip Input Connectivity	1529
13-50. DCAEVT1 Event Triggering	1532
13-51. DCAEVT2 Event Triggering	1532
13-52. DCBEVT1 Event Triggering	1533
13-53. DCBEVT2 Event Triggering	1533
13-54. Event Filtering	1534
13-55. Blanking Window Timing Diagram	1535
13-56. EPWM X-BAR	1536
13-57. EPWM Architecture - Single Output	1537
13-58. Simplified ePWM Module	1539
13-59. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave	1540
13-60. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$	1541
13-61. Buck Waveforms for (Note: Only three bucks shown here)	1542
13-62. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$)	1543
13-63. Buck Waveforms for (Note: $F_{PWM2} = F_{PWM1}$)	1544
13-64. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$)	1545
13-65. Half-H Bridge Waveforms for (Note: Here $F_{PWM2} = F_{PWM1}$)	1546

13-66. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control	1547
13-67. 3-Phase Inverter Waveforms for (Only One Inverter Shown)	1548
13-68. Configuring Two PWM Modules for Phase Control.....	1549
13-69. Timing Waveforms Associated With Phase Control Between Two Modules	1550
13-70. Control of a 3-Phase Interleaved DC/DC Converter	1551
13-71. 3-Phase Interleaved DC/DC Converter Waveforms for	1552
13-72. Controlling a Full-H Bridge Stage ($F_{PWM2} = F_{PWM1}$)	1553
13-73. ZVS Full-H Bridge Waveforms	1554
13-74. Peak Current Mode Control of a Buck Converter	1555
13-75. Peak Current Mode Control Waveforms for	1555
13-76. Control of Two Resonant Converter Stages	1556
13-77. H-Bridge LLC Resonant Converter PWM Waveforms.....	1556
13-78. TBCTL Register	1560
13-79. TBCTL2 Register	1562
13-80. TBCTR Register	1563
13-81. TBSTS Register	1564
13-82. CMPCTL Register	1565
13-83. CMPCTL2 Register	1567
13-84. DBCTL Register	1569
13-85. DBCTL2 Register.....	1572
13-86. AQCTL Register	1573
13-87. AQTSRCSEL Register.....	1575
13-88. PCCTL Register	1576
13-89. HRCNFG Register	1578
13-90. HRPWR Register	1580
13-91. HRMSTEP Register.....	1581
13-92. HRCNFG2 Register	1582
13-93. HRPCTL Register	1583
13-94. GLDCTL Register	1585
13-95. GLDCFG Register.....	1587
13-96. EPWMXLINK Register.....	1589
13-97. AQCTLA Register	1593
13-98. AQCTLA2 Register.....	1595
13-99. AQCTLB Register	1596
13-100. AQCTLB2 Register	1598
13-101. AQSFRC Register	1599
13-102. AQCSFRC Register	1600
13-103. DBREDHR Register	1601
13-104. DBRED Register	1602
13-105. DBFEDHR Register	1603
13-106. DBFED Register	1604
13-107. TBPHS Register.....	1605
13-108. TBPRDHR Register	1606
13-109. TBPRD Register	1607
13-110. CMPA Register.....	1608
13-111. CMPB Register.....	1609
13-112. CMPC Register.....	1610
13-113. CMPD Register.....	1611
13-114. GLDCTL2 Register	1612

13-115. TZSEL Register	1613
13-116. TZDCSEL Register	1615
13-117. TZCTL Register	1617
13-118. TZCTL2 Register.....	1619
13-119. TZCTLDCA Register	1621
13-120. TZCTLDCB Register	1623
13-121. TZEINT Register	1625
13-122. TZFLG Register	1626
13-123. TZCBCFLG Register	1628
13-124. TZOSTFLG Register	1629
13-125. TZCLR Register	1630
13-126. TZCBCCLR Register	1631
13-127. TZOSTCLR Register	1632
13-128. TZFRC Register.....	1633
13-129. ETSEL Register	1634
13-130. ETPS Register	1637
13-131. ETFLG Register.....	1640
13-132. ETCLR Register.....	1641
13-133. ETFRC Register.....	1642
13-134. ETINTPS Register	1643
13-135. ETSOCPS Register.....	1644
13-136. ETCNTINITCTL Register	1646
13-137. ETCNTINIT Register	1647
13-138. DCTRIPESEL Register	1648
13-139. DCACTL Register.....	1650
13-140. DCBCTL Register.....	1651
13-141. DCFCTL Register	1652
13-142. DCCAPCTL Register	1653
13-143. DCFOFFSET Register	1654
13-144. DCFOFFSETCNT Register.....	1655
13-145. DCFWINDOW Register	1656
13-146. DCFWINDOWCNT Register.....	1657
13-147. DCCAP Register	1658
13-148. DCAHTRIPSEL Register.....	1659
13-149. DCALTRIPSEL Register	1661
13-150. DCBHTRIPSEL Register.....	1663
13-151. DCBLTRIPSEL Register	1665
13-152. TRIP4MUX0TO15CFG Register	1668
13-153. TRIP4MUX16TO31CFG Register	1671
13-154. TRIP5MUX0TO15CFG Register	1674
13-155. TRIP5MUX16TO31CFG Register	1677
13-156. TRIP7MUX0TO15CFG Register	1680
13-157. TRIP7MUX16TO31CFG Register	1683
13-158. TRIP8MUX0TO15CFG Register	1686
13-159. TRIP8MUX16TO31CFG Register	1689
13-160. TRIP9MUX0TO15CFG Register	1692
13-161. TRIP9MUX16TO31CFG Register	1695
13-162. TRIP10MUX0TO15CFG Register	1698
13-163. TRIP10MUX16TO31CFG Register	1701

13-164. TRIP11MUX0TO15CFG Register	1704
13-165. TRIP11MUX16TO31CFG Register	1707
13-166. TRIP12MUX0TO15CFG Register	1710
13-167. TRIP12MUX16TO31CFG Register	1713
13-168. TRIP4MUXENABLE Register	1716
13-169. TRIP5MUXENABLE Register	1720
13-170. TRIP7MUXENABLE Register	1724
13-171. TRIP8MUXENABLE Register	1728
13-172. TRIP9MUXENABLE Register	1732
13-173. TRIP10MUXENABLE Register	1736
13-174. TRIP11MUXENABLE Register	1740
13-175. TRIP12MUXENABLE Register	1744
13-176. TRIPOUTINV Register	1748
13-177. TRIPLOCK Register	1750
13-178. SYNCSELECT Register	1752
13-179. EXTADCSOCSELECT Register	1754
13-180. SYNCSOCLOCK Register	1757
14-1. Resolution Calculations for Conventionally Generated PWM	1759
14-2. Operating Logic Using MEP	1761
14-3. HRPWM Extension Registers and Memory Configuration	1762
14-4. HRPWM System Interface	1763
14-5. HRPWM Block Diagram	1764
14-6. Required PWM Waveform for a Requested Duty = 40.5%	1766
14-7. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)	1769
14-8. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)	1771
14-9. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)	1771
14-10. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)	1772
14-11. Simple Buck Controlled Converter Using a Single PWM	1777
14-12. PWM Waveform Generated for Simple Buck Controlled Converter	1777
14-13. Simple Reconstruction Filter for a PWM-based DAC	1779
14-14. PWM Waveform Generated for the PWM DAC Function	1779
15-1. Capture and APWM Modes of Operation	1787
15-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode	1788
15-3. Capture Function Diagram	1789
15-4. Event Prescale Control	1790
15-5. Prescale Function Waveforms	1790
15-6. Details of the Continuous/One-shot Block	1791
15-7. Details of the Counter and Synchronization Block	1792
15-8. Time-Base Counter Synchronization Scheme 4	1793
15-9. Interrupts in eCAP Module	1794
15-10. PWM Waveform Details Of APWM Mode Operation	1795
15-11. Capture Sequence for Absolute Time-stamp and Rising Edge Detect	1797
15-12. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect	1799
15-13. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect	1801
15-14. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect	1803
15-15. PWM Waveform Details of APWM Mode Operation	1805
15-16. TSCTR Register	1809
15-17. CTRPHS Register	1810
15-18. CAP1 Register	1811

15-19. CAP2 Register	1812
15-20. CAP3 Register	1813
15-21. CAP4 Register	1814
15-22. ECCTL1 Register.....	1815
15-23. ECCTL2 Register.....	1817
15-24. ECEINT Register	1819
15-25. ECFLG Register	1820
15-26. ECCLR Register	1821
15-27. ECFRC Register.....	1822
16-1. Optical Encoder Disk	1824
16-2. QEP Encoder Output Signal for Forward/Reverse Movement	1824
16-3. Index Pulse Example	1825
16-4. Functional Block Diagram of the eQEP Peripheral	1827
16-5. Functional Block Diagram of Decoder Unit	1829
16-6. Quadrature Decoder State Machine	1830
16-7. Quadrature-clock and Direction Decoding	1831
16-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F).....	1833
16-9. Position Counter Underflow/Overflow (QPOSMAX = 4)	1834
16-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)	1836
16-11. Strobe Event Latch (QEPCTL[SEL] = 1)	1836
16-12. eQEP Position-compare Unit	1837
16-13. eQEP Position-compare Event Generation Points.....	1838
16-14. eQEP Position-compare Sync Output Pulse Stretcher.....	1838
16-15. eQEP Edge Capture Unit	1840
16-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)	1840
16-17. eQEP Edge Capture Unit - Timing Details	1841
16-18. eQEP Watchdog Timer.....	1842
16-19. eQEP Unit Time Base	1842
16-20. EQEP Interrupt Generation	1843
16-21. QPOSCNT Register.....	1845
16-22. QPOSINIT Register	1846
16-23. QPOSMAX Register	1847
16-24. QPOSCMP Register	1848
16-25. QPOSILAT Register	1849
16-26. QPOSSLAT Register	1850
16-27. QPOSLAT Register	1851
16-28. QUTMR Register	1852
16-29. QUPRD Register	1853
16-30. QWDTMR Register	1854
16-31. QWDPRD Register.....	1855
16-32. QDECCTL Register	1856
16-33. QEPCTL Register	1857
16-34. QCAPCTL Register	1860
16-35. QPOSCTL Register	1861
16-36. QEINT Register.....	1862
16-37. QFLG Register.....	1863
16-38. QCLR Register	1865
16-39. QFRC Register	1867
16-40. QEPSTS Register	1868

16-41. QCTMR Register	1869
16-42. QCPRD Register	1870
16-43. QCTMRLAT Register	1871
16-44. QCPRDLAT Register	1872
17-1. SPI CPU Interface.....	1874
17-2. SPI Master/Slave Connection	1877
17-3. SPI Module Master Configuration	1878
17-4. SPI Module Slave Configuration	1879
17-5. SPICLK Signal Options.....	1882
17-6. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1	1882
17-7. Five Bits per Character	1884
17-8. SPI DMA Trigger Diagram	1885
17-9. SPI Interrupt Flags and Enable Logic Generation	1886
17-10. SPI 3-wire Master Mode.....	1890
17-11. SPI 3-wire Slave Mode	1890
17-12. SPI Digital Audio Receiver Configuration Using 2 SPIs.....	1891
17-13. Standard Right-Justified Digital Audio Data Format	1891
17-14. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)	1892
17-15. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.)	1893
17-16. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.)	1894
17-17. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)	1895
17-18. SPISTE Behavior in Master Mode (Master lowers SPISTE during the entire 16 bits of transmission.)	1896
17-19. SPISTE Behavior in Slave Mode (Slave's SPISTE is lowered during the entire 16 bits of transmission.) ..	1897
17-20. SPICCR Register.....	1900
17-21. SPICTL Register	1902
17-22. SPISTS Register	1904
17-23. SPIBRR Register	1906
17-24. SPIRXEMU Register	1907
17-25. SPIRXBUF Register	1908
17-26. SPITXBUF Register.....	1909
17-27. SPIDAT Register	1910
17-28. SPIFFTX Register.....	1911
17-29. SPIFFRX Register.....	1913
17-30. SPIFFCT Register.....	1915
17-31. SPIPRI Register	1916
18-1. SCI CPU Interface	1918
18-2. Serial Communications Interface (SCI) Module Block Diagram	1919
18-3. Typical SCI Data Frame Formats	1921
18-4. Idle-Line Multiprocessor Communication Format	1923
18-5. Double-Buffered WUT and TXSHF	1923
18-6. Address-Bit Multiprocessor Communication Format.....	1925
18-7. SCI Asynchronous Communications Format	1925
18-8. SCI RX Signals in Communication Modes.....	1926
18-9. SCI TX Signals in Communications Mode	1926
18-10. SCI FIFO Interrupt Flags and Enable Logic	1929
18-11. SCICCR Register.....	1933

18-12. SCICTL1 Register	1935
18-13. SCIHBAUD Register	1937
18-14. SCILBAUD Register	1938
18-15. SCICTL2 Register	1939
18-16. SCIRXST Register	1940
18-17. SCIRXEMU Register	1942
18-18. SCIRXBUF Register	1943
18-19. SCITXBUF Register	1944
18-20. SCIFFTX Register	1945
18-21. SCIFFRX Register	1947
18-22. SCIFFCT Register	1949
18-23. SCIPRI Register	1950
19-1. Multiple I2C Modules Connected	1952
19-2. I2C Module Conceptual Block Diagram	1954
19-3. Clocking Diagram for the I2C Module	1954
19-4. The Roles of the Clock Divide-Down Values (ICCL and ICCH)	1955
19-5. Bit Transfer on the I2C-Bus	1956
19-6. I2C Module START and STOP Conditions	1958
19-7. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)	1958
19-8. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)	1958
19-9. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)	1959
19-10. I2C Module Free Data Format (FDF = 1 in I2CMDR)	1959
19-11. Repeated START Condition (in This Case, 7-Bit Addressing Format)	1960
19-12. Synchronization of Two I2C Clock Generators During Arbitration	1961
19-13. Arbitration Procedure Between Two Master-Transmitters	1962
19-14. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit	1962
19-15. Enable Paths of the I2C Interrupt Requests	1964
19-16. I2COAR Register	1968
19-17. I2CIER Register	1969
19-18. I2CSTR Register	1970
19-19. I2CCLKL Register	1974
19-20. I2CCLKH Register	1975
19-21. I2CCNT Register	1976
19-22. I2CDRR Register	1977
19-23. I2CSAR Register	1978
19-24. I2CDXR Register	1979
19-25. I2CMDR Register	1980
19-26. I2CISRC Register	1984
19-27. I2CEMDR Register	1985
19-28. I2CPSC Register	1986
19-29. I2CFFTX Register	1987
19-30. I2CFFRX Register	1988
20-1. Conceptual Block Diagram of the McBSP	1992
20-2. McBSP Data Transfer Paths	1993
20-3. Companding Processes	1994
20-4. μ -Law Transmit Data Companding Format	1994
20-5. A-Law Transmit Data Companding Format	1994
20-6. Two Methods by Which the McBSP Can Compand Internal Data	1995
20-7. Example - Clock Signal Control of Bit Transfer Timing	1995

20-8.	McBSP Operating at Maximum Packet Frequency	1997
20-9.	Single-Phase Frame for a McBSP Data Transfer.....	1998
20-10.	Dual-Phase Frame for a McBSP Data Transfer.....	1999
20-11.	Implementing the AC97 Standard With a Dual-Phase Frame.....	1999
20-12.	Timing of an AC97-Standard Data Transfer Near Frame Synchronization	2000
20-13.	McBSP Reception Physical Data Path	2000
20-14.	McBSP Reception Signal Activity	2000
20-15.	McBSP Transmission Physical Data Path	2001
20-16.	McBSP Transmission Signal Activity	2001
20-17.	Conceptual Block Diagram of the Sample Rate Generator.....	2003
20-18.	Possible Inputs to the Sample Rate Generator and the Polarity Bits	2005
20-19.	CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1.....	2007
20-20.	CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3.....	2008
20-21.	Overflow in the McBSP Receiver	2010
20-22.	Overflow Prevented in the McBSP Receiver	2011
20-23.	Possible Responses to Receive Frame-Synchronization Pulses	2011
20-24.	An Unexpected Frame-Synchronization Pulse During a McBSP Reception	2012
20-25.	Proper Positioning of Frame-Synchronization Pulses	2013
20-26.	Data in the McBSP Transmitter Overwritten and Thus Not Transmitted.....	2013
20-27.	Underflow During McBSP Transmission	2014
20-28.	Underflow Prevented in the McBSP Transmitter	2015
20-29.	Possible Responses to Transmit Frame-Synchronization Pulses	2015
20-30.	An Unexpected Frame-Synchronization Pulse During a McBSP Transmission	2016
20-31.	Proper Positioning of Frame-Synchronization Pulses	2017
20-32.	Alternating Between the Channels of Partition A and the Channels of Partition B	2019
20-33.	Reassigning Channel Blocks Throughout a McBSP Data Transfer	2020
20-34.	McBSP Data Transfer in the 8-Partition Mode	2021
20-35.	Activity on McBSP Pins for the Possible Values of XMCM.....	2024
20-36.	Typical SPI Interface.....	2025
20-37.	SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0	2027
20-38.	SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1	2027
20-39.	SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0	2027
20-40.	SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1	2027
20-41.	SPI Interface with McBSP Used as Master	2029
20-42.	SPI Interface With McBSP Used as Slave	2030
20-43.	Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0.....	2037
20-44.	Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1.....	2037
20-45.	Companding Processes for Reception and for Transmission	2038
20-46.	Range of Programmable Data Delay	2039
20-47.	2-Bit Data Delay Used to Skip a Framing Bit.....	2039
20-48.	Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..	2044
20-49.	Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods.....	2045
20-50.	Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..	2047
20-51.	Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0	2057
20-52.	Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1	2057
20-53.	Companding Processes for Reception and for Transmission	2058
20-54.	μ -Law Transmit Data Companding Format	2058
20-55.	A-Law Transmit Data Companding Format.....	2059
20-56.	Range of Programmable Data Delay	2060

20-57. 2-Bit Data Delay Used to Skip a Framing Bit.....	2060
20-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..	2064
20-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods.....	2064
20-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge ..	2066
20-61. Four 8-Bit Data Words Transferred To/From the McBSP.....	2070
20-62. One 32-Bit Data Word Transferred To/From the McBSP	2070
20-63. 8-Bit Data Words Transferred at Maximum Packet Frequency	2071
20-64. Configuring the Data Stream of as a Continuous 32-Bit Word.....	2071
20-65. Receive Interrupt Generation	2072
20-66. Transmit Interrupt Generation	2072
20-67. McBSP Interrupt Enable Register (MFFINT)	2073
20-68. Data Receive Registers (DRR2 and DRR1)	2076
20-69. Data Transmit Registers (DXR2 and DXR1)	2076
20-70. Serial Port Control 1 Register (SPCR1)	2077
20-71. Serial Port Control 2 Register (SPCR2)	2080
20-72. Receive Control Register 1 (RCR1)	2082
20-73. Receive Control Register 2 (RCR2)	2083
20-74. Transmit Control 1 Register (XCR1).....	2085
20-75. Transmit Control 2 Register (XCR2)	2086
20-76. Sample Rate Generator 1 Register (SRGR1)	2088
20-77. Sample Rate Generator 2 Register (SRGR2)	2088
20-78. Multichannel Control 1 Register (MCR1)	2090
20-79. Multichannel Control 2 Register (MCR2).....	2092
20-80. Pin Control Register (PCR)	2094
20-81. Receive Channel Enable Registers (RCERA...RCERH)	2096
20-82. Transmit Channel Enable Registers (XCERA...XCERH)	2098
21-1. CAN Block Diagram.....	2103
21-2. CAN Core in Silent Mode	2106
21-3. CAN Core in Loopback Mode	2106
21-4. CAN Core in External Loopback Mode.....	2107
21-5. CAN Core in Loopback Combined with Silent Mode	2108
21-6. Initialization of a Transmit Object	2112
21-7. Initialization of a single Receive Object for Data Frames	2112
21-8. Initialization of a single Receive Object for Remote Frames	2113
21-9. CPU Handling of a FIFO Buffer (Interrupt Driven)	2118
21-10. Bit Timing.....	2119
21-11. The Propagation Time Segment	2120
21-12. Synchronization on Late and Early Edges	2122
21-13. Filtering of Short Dominant Spikes.....	2123
21-14. Structure of the CAN Core's CAN Protocol Controller	2124
21-15. Data Transfer Between IF1 / IF2 Registers and Message RAM	2127
21-16. Structure of a Message Object	2128
21-17. Message RAM Representation in Debug Mode.....	2132
21-18. CAN_CTL Register.....	2135
21-19. CAN_ES Register	2137
21-20. CAN_ERRC Register	2139
21-21. CAN_BTR Register	2140
21-22. CAN_INT Register	2141
21-23. CAN_TEST Register.....	2142

21-24. CAN_PERR Register	2143
21-25. CAN_REL Register	2144
21-26. CAN_RAM_INIT Register	2145
21-27. CAN_GLB_INT_EN Register	2146
21-28. CAN_GLB_INT_FLG Register	2147
21-29. CAN_GLB_INT_CLR Register	2148
21-30. CAN_ABOTR Register	2149
21-31. CAN_TXRQ_X Register	2150
21-32. CAN_TXRQ_21 Register	2151
21-33. CAN_NDAT_X Register	2152
21-34. CAN_NDAT_21 Register	2153
21-35. CAN_IPEN_X Register	2154
21-36. CAN_IPEN_21 Register	2155
21-37. CAN_MVAL_X Register	2156
21-38. CAN_MVAL_21 Register	2157
21-39. CAN_IP_MUX21 Register	2158
21-40. CAN_IF1CMD Register	2159
21-41. CAN_IF1MSK Register	2162
21-42. CAN_IF1ARB Register	2163
21-43. CAN_IF1MCTL Register	2164
21-44. CAN_IF1DATA Register	2166
21-45. CAN_IF1DATB Register	2167
21-46. CAN_IF2CMD Register	2168
21-47. CAN_IF2MSK Register	2171
21-48. CAN_IF2ARB Register	2172
21-49. CAN_IF2MCTL Register	2173
21-50. CAN_IF2DATA Register	2175
21-51. CAN_IF2DATB Register	2176
21-52. CAN_IF3OBS Register	2177
21-53. CAN_IF3MSK Register	2179
21-54. CAN_IF3ARB Register	2180
21-55. CAN_IF3MCTL Register	2181
21-56. CAN_IF3DATA Register	2183
21-57. CAN_IF3DATB Register	2184
21-58. CAN_IF3UPD Register	2185
22-1. USB Block Diagram	2187
22-2. USB Scheme	2188
22-3. Function Address Register (USBFADDR)	2204
22-4. Power Management Register (USBPOWER) in Host Mode	2205
22-5. Power Management Register (USBPOWER) in Device Mode	2205
22-6. USB Transmit Interrupt Status Register (USBTXIS)	2207
22-7. USB Receive Interrupt Status Register (USBRXIS)	2208
22-8. USB Transmit Interrupt Status Enable Register (USBTXIE)	2209
22-9. USB Receive Interrupt Enable Register (USBRXIE)	2210
22-10. USB General Interrupt Status Register (USBIS) in Host Mode	2211
22-11. USB General Interrupt Status Register (USBIS) in Device Mode	2212
22-12. USB Interrupt Enable Register (USBIE) in Host Mode	2213
22-13. USB Interrupt Enable Register (USBIE) in Device Mode	2214
22-14. Frame Number Register (FRAME)	2215

22-15. USB Endpoint Index Register (USBEPIDX)	2215
22-16. USB Test Mode Register (USBTEST) in Host Mode	2216
22-17. USB Test Mode Register (USBTEST) in Device Mode	2216
22-18. USB FIFO Endpoint <i>n</i> Register (USBFIFO[<i>n</i>])	2218
22-19. USB Device Control Register (USBDEVCTL)	2219
22-20. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ)	2221
22-21. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ)	2222
22-22. USB Transmit FIFO Start Address Register (USBTXFIFOADDR)	2223
22-23. USB Receive FIFO Start Address Register (USBRXFIFOADDR)	2224
22-24. USB Connect Timing Register (USBCONTIM)	2225
22-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF)	2226
22-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF)	2226
22-27. USB Transmit Functional Address Endpoint <i>n</i> Registers (USBTXFUNCADDR[<i>n</i>])	2227
22-28. USB Transmit Hub Address Endpoint <i>n</i> Registers (USBTXHUBADDR[<i>n</i>])	2228
22-29. USB Transmit Hub Port Endpoint <i>n</i> Registers (USBTXHUBPORT[<i>n</i>])	2229
22-30. USB Receive Functional Address Endpoint <i>n</i> Registers (USBFIFO[<i>n</i>])	2230
22-31. USB Receive Hub Address Endpoint <i>n</i> Registers (USBRXHUBADDR[<i>n</i>])	2231
22-32. USB Transmit Hub Port Endpoint <i>n</i> Registers (USBRXHUBPORT[<i>n</i>])	2232
22-33. USB Maximum Transmit Data Endpoint <i>n</i> Registers (USBTXMAXP[<i>n</i>])	2233
22-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode	2234
22-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode	2235
22-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode	2236
22-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode	2236
22-38. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0)	2237
22-39. USB Type Endpoint 0 Register (USBTYPE0)	2237
22-40. USB NAK Limit Register (USBNAKLMT)	2238
22-41. USB Transmit Control and Status Endpoint <i>n</i> Low Register (USBTXCSSL[<i>n</i>]) in Host Mode	2239
22-42. USB Transmit Control and Status Endpoint <i>n</i> Low Register (USBTXCSSL[<i>n</i>]) in Device Mode	2240
22-43. USB Transmit Control and Status Endpoint <i>n</i> High Register (USBTXCSSH[<i>n</i>]) in Host Mode	2242
22-44. USB Transmit Control and Status Endpoint <i>n</i> High Register (USBTXCSSH[<i>n</i>]) in Device Mode	2243
22-45. USB Maximum Receive Data Endpoint <i>n</i> Registers (USBRXMAXP[<i>n</i>])	2244
22-46. USB Receive Control and Status Endpoint <i>n</i> Low Register (USBCSSL[<i>n</i>]) in Host Mode	2245
22-47. USB Control and Status Endpoint <i>n</i> Low Register (USBCSSL[<i>n</i>]) in Device Mode	2246
22-48. USB Receive Control and Status Endpoint <i>n</i> High Register (USBCSSH[<i>n</i>]) in Host Mode	2248
22-49. USB Control and Status Endpoint <i>n</i> High Register (USBCSSH[<i>n</i>]) in Device Mode	2249
22-50. USB Maximum Receive Data Endpoint <i>n</i> Registers (USBRXCOUNT[<i>n</i>])	2250
22-51. USB Host Transmit Configure Type Endpoint <i>n</i> Register (USBTXTYPE[<i>n</i>])	2251
22-52. USB Host Transmit Interval Endpoint <i>n</i> Register (USBTXINTERVAL[<i>n</i>])	2252
22-53. USB Host Configure Receive Type Endpoint <i>n</i> Register (USBRXTYPE[<i>n</i>])	2253
22-54. USB Host Receive Polling Interval Endpoint <i>n</i> Register (USBRXINTERVAL[<i>n</i>])	2254
22-55. USB Request Packet Count in Block Transfer Endpoint <i>n</i> Registers (USBRQPKTCOUNT[<i>n</i>])	2255
22-56. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUDIS)	2256
22-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUDIS)	2257
22-58. USB External Power Control Register (USBEPIC)	2258
22-59. USB External Power Control Raw Interrupt Status Register (USBEPICRIS)	2260
22-60. USB External Power Control Interrupt Mask Register (USBEPICIM)	2261
22-61. USB External Power Control Interrupt Status and Clear Register (USBEPICISC)	2262
22-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)	2263
22-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)	2264

22-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC).....	2265
22-65. USB General-Purpose Control and Status Register (USBGPCS)	2266
22-66. USB DMA Select Register (USBDMASEL)	2267
23-1. uPP Integration	2270
23-2. Functional Block Diagram	2271
23-3. RX in SDR or DDR (non-demux) Mode	2272
23-4. RX in DDR (demux) Mode	2272
23-5. TX in SDR (non-interleave) or DDR (non-demux) Mode.....	2272
23-6. TX in SDR (interleave) or DDR (demux) Mode	2272
23-7. IO Output Clock Generation for TX Mode.....	2273
23-8. IO Input clock for RX Mode	2273
23-9. Structure of DMA Window and Lines in Memory.....	2275
23-10. uPP Receive in SDR Mode	2277
23-11. uPP Transmit in SDR Mode	2278
23-12. uPP Transmit in SDR Mode – Interleaving	2278
23-13. uPP Receive DDR Case	2278
23-14. uPP Transmit DDR Case.....	2278
23-15. uPP Tx Data Pattern in Non-Interleaved Mode	2279
23-16. uPP Rx Data Pattern in Non-Interleaved Mode	2279
23-17. PID Register	2286
23-18. PERCTL Register	2287
23-19. CHCTL Register	2288
23-20. IFCFG Register.....	2289
23-21. IFIVAL Register.....	2291
23-22. THCFG Register.....	2292
23-23. RAWINTST Register	2294
23-24. ENINTST Register	2296
23-25. INTENSET Register.....	2298
23-26. INTENCLR Register	2300
23-27. CHIDESC0 Register	2302
23-28. CHIDESC1 Register	2303
23-29. CHIDESC2 Register	2304
23-30. CHIST0 Register	2305
23-31. CHIST1 Register	2306
23-32. CHIST2 Register	2307
23-33. CHQDESC0 Register.....	2308
23-34. CHQDESC1 Register	2309
23-35. CHQDESC2 Register	2310
23-36. CHQST0 Register	2311
23-37. CHQST1 Register	2312
23-38. CHQST2 Register	2313
23-39. GINTEN Register.....	2314
23-40. GINTFLG Register	2315
23-41. GINTCLR Register	2316
23-42. DLYCTL Register.....	2317
24-1. EMIF Module Overview.....	2319
24-2. EMIF Functional Block Diagram.....	2321
24-3. Timing Waveform of SDRAM PRE Command	2325
24-4. EMIF to 2M x 16 x 4 bank SDRAM Interface	2325

24-5. EMIF to 512K × 16 × 2 bank SDRAM Interface	2326
24-6. Timing Waveform for Basic SDRAM Read Operation	2333
24-7. Timing Waveform for Basic SDRAM Write Operation	2334
24-8. EMIF Asynchronous Interface	2336
24-9. EMIF to 8-bit/16-bit Memory Interface	2337
24-10. Common Asynchronous Interface	2337
24-11. Timing Waveform of an Asynchronous Read Cycle in Normal Mode	2341
24-12. Timing Waveform of an Asynchronous Write Cycle in Normal Mode	2343
24-13. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode	2345
24-14. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode	2347
24-15. Example Configuration Interface	2355
24-16. SDRAM Timing Register (SDRAM_TR)	2356
24-17. SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)	2357
24-18. SDRAM Refresh Control Register (SDRAM_RCR)	2357
24-19. SDRAM Configuration Register (SDRAM_CR)	2358
24-20. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms	2359
24-21. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms	2360
24-22. Asynchronous <i>m</i> Configuration Register (<i>m</i> = 1, 2) (ASYNC_CS _{<i>n</i>} _CR(<i>n</i> = 2, 3))	2361
24-23. RCSR Register	2364
24-24. ASYNC_WCCR Register	2365
24-25. SDRAM_CR Register	2366
24-26. SDRAM_RCR Register	2368
24-27. ASYNC_CS2_CR Register	2369
24-28. ASYNC_CS3_CR Register	2370
24-29. ASYNC_CS4_CR Register	2371
24-30. SDRAM_TR Register	2372
24-31. TOTAL_SDRAM_AR Register	2373
24-32. TOTAL_SDRAM_ACTR Register	2374
24-33. SDR_EXT_TMNG Register	2375
24-34. INT_RAW Register	2376
24-35. INT_MSK Register	2377
24-36. INT_MSK_SET Register	2378
24-37. INT_MSK_CLR Register	2379
24-38. EMIF1LOCK Register	2381
24-39. EMIF1COMMIT Register	2382
24-40. EMIF1MSEL Register	2383
24-41. EMIF1ACCPROT0 Register	2384
24-42. EMIF2LOCK Register	2386
24-43. EMIF2COMMIT Register	2387
24-44. EMIF2ACCPROT0 Register	2388

List of Tables

1-1.	TMU Supported Instructions	80
1-2.	Viterbi Decode Performance	81
1-3.	Complex Math Performance	81
2-1.	Reset Signals	84
2-2.	PIE Channel Mapping	91
2-3.	CPU Interrupt Vectors	92
2-4.	PIE Interrupt Vectors.....	93
2-5.	Access to EALLOW-Protected Registers	100
2-6.	Clock Connections Sorted by Clock Domain.....	108
2-7.	Clock Connections Sorted by Module Name.....	109
2-8.	Example Watchdog Key Sequences	114
2-9.	Local Shared RAM.....	120
2-10.	Global Shared RAM	121
2-11.	Error Handling in Different Scenarios	126
2-12.	Mapping of ECC Bits in Read Data from ECC/Parity Address Map	127
2-13.	Mapping of Parity Bits in Read Data from ECC/Parity Address Map	127
2-14.	RAM Status	140
2-15.	Security Levels	141
2-16.	System Control Base Address Table.....	152
2-17.	CPUTIMER_REGS Registers	153
2-18.	TIM Register Field Descriptions	154
2-19.	PRD Register Field Descriptions	155
2-20.	TCR Register Field Descriptions.....	156
2-21.	TPR Register Field Descriptions.....	158
2-22.	TPRH Register Field Descriptions.....	159
2-23.	PIE_CTRL_REGS Registers	160
2-24.	PIECTRL Register Field Descriptions	161
2-25.	PIEACK Register Field Descriptions	162
2-26.	PIEIER1 Register Field Descriptions.....	163
2-27.	PIEIFR1 Register Field Descriptions	164
2-28.	PIEIER2 Register Field Descriptions.....	165
2-29.	PIEIFR2 Register Field Descriptions	166
2-30.	PIEIER3 Register Field Descriptions.....	167
2-31.	PIEIFR3 Register Field Descriptions	168
2-32.	PIEIER4 Register Field Descriptions.....	169
2-33.	PIEIFR4 Register Field Descriptions	170
2-34.	PIEIER5 Register Field Descriptions.....	171
2-35.	PIEIFR5 Register Field Descriptions	172
2-36.	PIEIER6 Register Field Descriptions.....	173
2-37.	PIEIFR6 Register Field Descriptions	174
2-38.	PIEIER7 Register Field Descriptions.....	175
2-39.	PIEIFR7 Register Field Descriptions	176
2-40.	PIEIER8 Register Field Descriptions.....	177
2-41.	PIEIFR8 Register Field Descriptions	178
2-42.	PIEIER9 Register Field Descriptions.....	179
2-43.	PIEIFR9 Register Field Descriptions	180
2-44.	PIEIER10 Register Field Descriptions	181

2-45.	PIEIFR10 Register Field Descriptions	182
2-46.	PIEIER11 Register Field Descriptions	183
2-47.	PIEIFR11 Register Field Descriptions	184
2-48.	PIEIER12 Register Field Descriptions	185
2-49.	PIEIFR12 Register Field Descriptions	186
2-50.	WD_REGS Registers	187
2-51.	SCSR Register Field Descriptions.....	188
2-52.	WDCNTR Register Field Descriptions	189
2-53.	WDKEY Register Field Descriptions	190
2-54.	WDCR Register Field Descriptions.....	191
2-55.	WDWCR Register Field Descriptions	192
2-56.	NMI_INTERRUPT_REGS Registers.....	193
2-57.	NMICFG Register Field Descriptions	194
2-58.	NMIFLG Register Field Descriptions	195
2-59.	NMIFLGCLR Register Field Descriptions.....	197
2-60.	NMIFLGFRC Register Field Descriptions	199
2-61.	NMIWDCNT Register Field Descriptions	201
2-62.	NMIWDPRD Register Field Descriptions	202
2-63.	NMISHDFLG Register Field Descriptions	203
2-64.	XINT_REGS Registers.....	205
2-65.	XINT1CR Register Field Descriptions.....	206
2-66.	XINT2CR Register Field Descriptions.....	207
2-67.	XINT3CR Register Field Descriptions.....	208
2-68.	XINT4CR Register Field Descriptions.....	209
2-69.	XINT5CR Register Field Descriptions.....	210
2-70.	XINT1CTR Register Field Descriptions	211
2-71.	XINT2CTR Register Field Descriptions	212
2-72.	XINT3CTR Register Field Descriptions	213
2-73.	DMA_CLA_SRC_SEL_REGS Registers	214
2-74.	CLA1TASKSRCSELLOCK Register Field Descriptions	215
2-75.	DMACHSRCSELLOCK Register Field Descriptions	216
2-76.	CLA1TASKSRCSEL1 Register Field Descriptions	217
2-77.	CLA1TASKSRCSEL2 Register Field Descriptions	218
2-78.	DMACHSRCSEL1 Register Field Descriptions	219
2-79.	DMACHSRCSEL2 Register Field Descriptions	220
2-80.	DEV_CFG_REGS Registers	221
2-81.	DEVCFGLOCK1 Register Field Descriptions	223
2-82.	PARTIDL Register Field Descriptions.....	225
2-83.	PARTIDH Register Field Descriptions	226
2-84.	REVID Register Field Descriptions	227
2-85.	DC0 Register Field Descriptions.....	228
2-86.	DC1 Register Field Descriptions.....	229
2-87.	DC2 Register Field Descriptions.....	230
2-88.	DC3 Register Field Descriptions.....	231
2-89.	DC4 Register Field Descriptions.....	233
2-90.	DC5 Register Field Descriptions.....	234
2-91.	DC6 Register Field Descriptions.....	235
2-92.	DC7 Register Field Descriptions.....	236
2-93.	DC8 Register Field Descriptions.....	237

2-94.	DC9 Register Field Descriptions	238
2-95.	DC10 Register Field Descriptions	239
2-96.	DC11 Register Field Descriptions	240
2-97.	DC12 Register Field Descriptions	241
2-98.	DC13 Register Field Descriptions	242
2-99.	DC14 Register Field Descriptions	243
2-100.	DC15 Register Field Descriptions	244
2-101.	DC17 Register Field Descriptions	245
2-102.	DC18 Register Field Descriptions	246
2-103.	DC19 Register Field Descriptions	247
2-104.	DC20 Register Field Descriptions	248
2-105.	PERCNF1 Register Field Descriptions.....	250
2-106.	FUSEERR Register Field Descriptions	251
2-107.	SOFTPRES0 Register Field Descriptions	252
2-108.	SOFTPRES1 Register Field Descriptions	253
2-109.	SOFTPRES2 Register Field Descriptions	254
2-110.	SOFTPRES3 Register Field Descriptions	256
2-111.	SOFTPRES4 Register Field Descriptions	257
2-112.	SOFTPRES6 Register Field Descriptions	258
2-113.	SOFTPRES7 Register Field Descriptions	259
2-114.	SOFTPRES8 Register Field Descriptions	260
2-115.	SOFTPRES9 Register Field Descriptions	261
2-116.	SOFTPRES11 Register Field Descriptions.....	262
2-117.	SOFTPRES13 Register Field Descriptions.....	263
2-118.	SOFTPRES14 Register Field Descriptions.....	264
2-119.	SOFTPRES16 Register Field Descriptions.....	265
2-120.	CPUSEL0 Register Field Descriptions	266
2-121.	CPUSEL1 Register Field Descriptions	268
2-122.	CPUSEL2 Register Field Descriptions	269
2-123.	CPUSEL3 Register Field Descriptions	270
2-124.	CPUSEL4 Register Field Descriptions	271
2-125.	CPUSEL5 Register Field Descriptions	272
2-126.	CPUSEL6 Register Field Descriptions	273
2-127.	CPUSEL7 Register Field Descriptions	274
2-128.	CPUSEL8 Register Field Descriptions	275
2-129.	CPUSEL9 Register Field Descriptions	276
2-130.	CPUSEL11 Register Field Descriptions	277
2-131.	CPUSEL12 Register Field Descriptions	279
2-132.	CPUSEL14 Register Field Descriptions	280
2-133.	CPU2RESCTL Register Field Descriptions	281
2-134.	RSTSTAT Register Field Descriptions	282
2-135.	LPMSTAT Register Field Descriptions.....	283
2-136.	CLK_CFG_REGS Registers	284
2-137.	CLKSEM Register Field Descriptions	285
2-138.	CLKCFGLOCK1 Register Field Descriptions	286
2-139.	CLKSRCCTL1 Register Field Descriptions.....	288
2-140.	CLKSRCCTL2 Register Field Descriptions.....	290
2-141.	CLKSRCCTL3 Register Field Descriptions.....	292
2-142.	SYSPLLCTL1 Register Field Descriptions	293

2-143. SYSPLLMULT Register Field Descriptions.....	294
2-144. SYSPLLSTS Register Field Descriptions.....	295
2-145. AUXPLLCTL1 Register Field Descriptions	296
2-146. AUXPLLMULT Register Field Descriptions	297
2-147. AUXPLLSTS Register Field Descriptions.....	298
2-148. SYSCLKDIVSEL Register Field Descriptions	299
2-149. AUXCLKDIVSEL Register Field Descriptions	300
2-150. PERCLKDIVSEL Register Field Descriptions	301
2-151. XCLKOUTDIVSEL Register Field Descriptions	302
2-152. LOSPCP Register Field Descriptions	303
2-153. MCDCCR Register Field Descriptions	304
2-154. X1CNT Register Field Descriptions	305
2-155. CPU_SYS_REGS Registers	306
2-156. CPUSYSLOCK1 Register Field Descriptions	307
2-157. HIBBOOTMODE Register Field Descriptions	310
2-158. IORESTOREADDR Register Field Descriptions.....	311
2-159. PIEVERRADDR Register Field Descriptions.....	312
2-160. PCLKCR0 Register Field Descriptions.....	313
2-161. PCLKCR1 Register Field Descriptions.....	315
2-162. PCLKCR2 Register Field Descriptions.....	316
2-163. PCLKCR3 Register Field Descriptions.....	318
2-164. PCLKCR4 Register Field Descriptions.....	319
2-165. PCLKCR6 Register Field Descriptions.....	320
2-166. PCLKCR7 Register Field Descriptions.....	321
2-167. PCLKCR8 Register Field Descriptions.....	322
2-168. PCLKCR9 Register Field Descriptions.....	323
2-169. PCLKCR10 Register Field Descriptions	324
2-170. PCLKCR11 Register Field Descriptions	325
2-171. PCLKCR12 Register Field Descriptions	326
2-172. PCLKCR13 Register Field Descriptions	327
2-173. PCLKCR14 Register Field Descriptions	328
2-174. PCLKCR16 Register Field Descriptions	329
2-175. SECMSEL Register Field Descriptions	330
2-176. LPMCR Register Field Descriptions.....	331
2-177. GPIOLPMSEL0 Register Field Descriptions	333
2-178. GPIOLPMSEL1 Register Field Descriptions	335
2-179. TMR2CLKCTL Register Field Descriptions	337
2-180. RESC Register Field Descriptions.....	338
2-181. ROM_PREFETCH_REGS Registers.....	340
2-182. ROMPREFETCH Register Field Descriptions	341
2-183. DCSM_Z1_OTP Registers	342
2-184. Z1OTP_LINKPOINTER1 Register Field Descriptions	343
2-185. Z1OTP_LINKPOINTER2 Register Field Descriptions	344
2-186. Z1OTP_LINKPOINTER3 Register Field Descriptions	345
2-187. Z1OTP_PSWDLOCK Register Field Descriptions	346
2-188. Z1OTP_CRCLOCK Register Field Descriptions.....	347
2-189. Z1OTP_BOOTCTRL Register Field Descriptions	348
2-190. DCSM_Z2_OTP Registers	349
2-191. Z2OTP_LINKPOINTER1 Register Field Descriptions	350

2-192. Z2OTP_LINKPOINTER2 Register Field Descriptions	351
2-193. Z2OTP_LINKPOINTER3 Register Field Descriptions	352
2-194. Z2OTP_PSWDLOCK Register Field Descriptions	353
2-195. Z2OTP_CRCLOCK Register Field Descriptions	354
2-196. Z2OTP_BOOTCTRL Register Field Descriptions	355
2-197. DCSM_Z1_REGS Registers	356
2-198. Z1_LINKPOINTER Register Field Descriptions	357
2-199. Z1_OTPSECLOCK Register Field Descriptions	358
2-200. Z1_BOOTCTRL Register Field Descriptions	359
2-201. Z1_LINKPOINTERERR Register Field Descriptions	360
2-202. Z1_CSMKEY0 Register Field Descriptions	361
2-203. Z1_CSMKEY1 Register Field Descriptions	362
2-204. Z1_CSMKEY2 Register Field Descriptions	363
2-205. Z1_CSMKEY3 Register Field Descriptions	364
2-206. Z1_CR Register Field Descriptions	365
2-207. Z1_GRABSECTR Register Field Descriptions	366
2-208. Z1_GRABRAMR Register Field Descriptions	369
2-209. Z1_EXEONLYSECTR Register Field Descriptions	371
2-210. Z1_EXEONLYRAMR Register Field Descriptions	374
2-211. DCSM_Z2_REGS Registers	376
2-212. Z2_LINKPOINTER Register Field Descriptions	377
2-213. Z2_OTPSECLOCK Register Field Descriptions	378
2-214. Z2_BOOTCTRL Register Field Descriptions	379
2-215. Z2_LINKPOINTERERR Register Field Descriptions	380
2-216. Z2_CSMKEY0 Register Field Descriptions	381
2-217. Z2_CSMKEY1 Register Field Descriptions	382
2-218. Z2_CSMKEY2 Register Field Descriptions	383
2-219. Z2_CSMKEY3 Register Field Descriptions	384
2-220. Z2_CR Register Field Descriptions	385
2-221. Z2_GRABSECTR Register Field Descriptions	386
2-222. Z2_GRABRAMR Register Field Descriptions	389
2-223. Z2_EXEONLYSECTR Register Field Descriptions	391
2-224. Z2_EXEONLYRAMR Register Field Descriptions	394
2-225. DCSM_COMMON_REGS Registers	396
2-226. FLSEM Register Field Descriptions	397
2-227. SECTSTAT Register Field Descriptions	398
2-228. RAMSTAT Register Field Descriptions	401
2-229. MEM_CFG_REGS Registers	403
2-230. DxLOCK Register Field Descriptions	404
2-231. DxCOMMIT Register Field Descriptions	405
2-232. DxACCPROT0 Register Field Descriptions	406
2-233. DxTEST Register Field Descriptions	407
2-234. DxINIT Register Field Descriptions	408
2-235. DxINITDONE Register Field Descriptions	409
2-236. LSxLOCK Register Field Descriptions	410
2-237. LSxCOMMIT Register Field Descriptions	411
2-238. LSxMSEL Register Field Descriptions	413
2-239. LSxCLAPGM Register Field Descriptions	415
2-240. LSxACCPROT0 Register Field Descriptions	416

2-241. LSxACCPROT1 Register Field Descriptions	418
2-242. LSxTEST Register Field Descriptions	419
2-243. LSxINIT Register Field Descriptions	421
2-244. LSxINITDONE Register Field Descriptions	422
2-245. GSxLOCK Register Field Descriptions	423
2-246. GSxCOMMIT Register Field Descriptions	425
2-247. GSxMSEL Register Field Descriptions	427
2-248. GSxACCPROT0 Register Field Descriptions	429
2-249. GSxACCPROT1 Register Field Descriptions	431
2-250. GSxACCPROT2 Register Field Descriptions	433
2-251. GSxACCPROT3 Register Field Descriptions	435
2-252. GSxTEST Register Field Descriptions	437
2-253. GSxINIT Register Field Descriptions	440
2-254. GSxINITDONE Register Field Descriptions	442
2-255. MSGxTEST Register Field Descriptions	444
2-256. MSGxINIT Register Field Descriptions	445
2-257. MSGxINITDONE Register Field Descriptions	446
2-258. ACCESS_PROTECTION_REGS Registers	447
2-259. NMAVFLG Register Field Descriptions	448
2-260. NMAVSET Register Field Descriptions	449
2-261. NMAVCLR Register Field Descriptions	450
2-262. NMAVINTEN Register Field Descriptions	451
2-263. NMCPURDAVADDR Register Field Descriptions	452
2-264. NMCPUWRAVADDR Register Field Descriptions	453
2-265. NMCPUFAVADDR Register Field Descriptions	454
2-266. NMDMAWRAVADDR Register Field Descriptions	455
2-267. NMCLA1RDAVADDR Register Field Descriptions	456
2-268. NMCLA1WRAVADDR Register Field Descriptions	457
2-269. NMCLA1FAVADDR Register Field Descriptions	458
2-270. MAVFLG Register Field Descriptions	459
2-271. MAVSET Register Field Descriptions	460
2-272. MAVCLR Register Field Descriptions	461
2-273. MAVINTEN Register Field Descriptions	462
2-274. MCPUFAVADDR Register Field Descriptions	463
2-275. MCPUWRAVADDR Register Field Descriptions	464
2-276. MDMAWRAVADDR Register Field Descriptions	465
2-277. MEMORY_ERROR_REGS Registers	466
2-278. UCERRFLG Register Field Descriptions	467
2-279. UCERRSET Register Field Descriptions	468
2-280. UCERRCLR Register Field Descriptions	469
2-281. UCCPUREADDR Register Field Descriptions	470
2-282. UCDMAREADDR Register Field Descriptions	471
2-283. UCCLA1READADDR Register Field Descriptions	472
2-284. CERRFLG Register Field Descriptions	473
2-285. CERRSET Register Field Descriptions	474
2-286. CERRCLR Register Field Descriptions	475
2-287. CCPUREADDR Register Field Descriptions	476
2-288. CERRCNT Register Field Descriptions	477
2-289. CERRTHRES Register Field Descriptions	478

2-290. CEINTFLG Register Field Descriptions	479
2-291. CEINTCLR Register Field Descriptions.....	480
2-292. CEINTSET Register Field Descriptions	481
2-293. CEINTEN Register Field Descriptions	482
2-294. ROM_WAIT_STATE_REGS Registers	483
2-295. ROMWAITSTATE Register Field Descriptions	484
2-296. FLASH_CTRL_REGS Registers.....	485
2-297. FRDCNTL Register Field Descriptions.....	486
2-298. FSPRD Register Field Descriptions	487
2-299. FBAC Register Field Descriptions	488
2-300. FBFALLBACK Register Field Descriptions	489
2-301. FBPRDY Register Field Descriptions	490
2-302. FPAC1 Register Field Descriptions	491
2-303. FPAC2 Register Field Descriptions	492
2-304. FMAC Register Field Descriptions	493
2-305. FMSTAT Register Field Descriptions	494
2-306. FRD_INTF_CTRL Register Field Descriptions.....	496
2-307. FLASH_ECC_REGS Registers	497
2-308. ECC_ENABLE Register Field Descriptions	498
2-309. SINGLE_ERR_ADDR_LOW Register Field Descriptions	499
2-310. SINGLE_ERR_ADDR_HIGH Register Field Descriptions.....	500
2-311. UNC_ERR_ADDR_LOW Register Field Descriptions	501
2-312. UNC_ERR_ADDR_HIGH Register Field Descriptions.....	502
2-313. ERR_STATUS Register Field Descriptions	503
2-314. ERR_POS Register Field Descriptions	505
2-315. ERR_STATUS_CLR Register Field Descriptions	506
2-316. ERR_CNT Register Field Descriptions	507
2-317. ERR_THRESHOLD Register Field Descriptions	508
2-318. ERR_INTFLG Register Field Descriptions	509
2-319. ERR_INTCLR Register Field Descriptions	510
2-320. FDATAH_TEST Register Field Descriptions	511
2-321. FDATA_L_TEST Register Field Descriptions	512
2-322. FADDR_TEST Register Field Descriptions	513
2-323. FECC_TEST Register Field Descriptions	514
2-324. FECC_CTRL Register Field Descriptions	515
2-325. FOUTH_TEST Register Field Descriptions	516
2-326. FOUTL_TEST Register Field Descriptions	517
2-327. FECC_STATUS Register Field Descriptions.....	518
3-1. Device Boot Mode – Decoded by CPU1.....	524
3-2. EMU_BOOTCTRL Definition: - @ 0xD00 for CPU1 (32 bit word).....	524
3-3. Emulation Boot (TRSTn == 1)	525
3-4. BOOTCTRL Register Bit Definition for CPU1	526
3-5. BOOTCTRL Register Bit Definition for CPU2	526
3-6. Get Mode Decoding on CPU1	527
3-7. Get Mode Decoding on CPU2	527
3-8. Stand Alone Boot Modes With TRSTn = 0 (on CPU1)	528
3-9. EMU_BOOTCTRL Definition: - @ 0xD00 for CPU2	528
3-10. Emulation Boot (TRSTn == 1) – CPU2	529
3-11. Stand Alone Boot Options on CPU2	529

3-12.	Reset Causes	534
3-13.	Exceptions Handling by Boot ROM	534
3-14.	C2TOC1IPC Commands Table	538
3-15.	C1TOC2IPC Commands Table	539
3-16.	Time Diagram	541
3-17.	ROM Wait States	545
3-18.	General Structure Of Source Program Data Stream In 16-Bit Mode	547
3-19.	LSB/MSB Loading Sequence in 8-Bit Data Stream	549
3-20.	SPI 8-Bit Data Stream	556
3-21.	I2C 8-Bit Data Stream	561
3-22.	Parallel GPIO Boot 8-Bit Data Stream	562
3-23.	Bit-Rate Value for Internal Oscillators.....	566
3-24.	CAN 8-Bit Data Stream	566
3-25.	USB 8-Bit Data Stream	568
3-26.	CLA Data ROM Tables.....	569
4-1.	Peripheral Interrupt Trigger Source Options	579
4-2.	DMA Register Summary	591
4-3.	DMA Control Register (DMACTRL) Field Descriptions	592
4-4.	Debug Control Register (DEBUGCTRL) Field Descriptions	593
4-5.	Revision Register (REVISION) Field Descriptions	593
4-6.	Priority Control Register 1 (PRIORITYCTRL1) Field Descriptions	594
4-7.	Priority Status Register (PRIORITYSTAT) Field Descriptions	595
4-8.	Mode Register (MODE) Field Descriptions.....	596
4-9.	Control Register (CONTROL) Field Descriptions.....	598
4-10.	Burst Size Register (BURST_SIZE) Field Descriptions.....	600
4-11.	Burst Count Register (BURST_COUNT) Field Descriptions	600
4-12.	Source Burst Step Size Register (SRC_BURST_STEP) Field Descriptions.....	601
4-13.	Destination Burst Step Register Size (DST_BURST_STEP) Field Descriptions	602
4-14.	Transfer Size Register (TRANSFER_SIZE) Field Descriptions	602
4-15.	Transfer Count Register (TRANSFER_COUNT) Field Descriptions	603
4-16.	Source Transfer Step Size Register (SRC_TRANSFER_STEP) Field Descriptions	603
4-17.	Destination Transfer Step Size Register (DST_TRANSFER_STEP) Field Descriptions.....	604
4-18.	Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) Field Descriptions	604
4-19.	Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) Field Descriptions	605
4-20.	Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) Field Descriptions.....	605
4-21.	Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) Field Descriptions	606
4-22.	Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR) Field Descriptions	606
4-23.	Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) Field Descriptions.....	607
4-24.	Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR) Field Descriptions	607
5-1.	Configuration Options	613
5-2.	Write Followed by Read - Read Occurs First	622
5-3.	Write Followed by Read - Write Occurs First	622
5-4.	ADC to CLA Early Interrupt Response	624
5-5.	Operand Nomenclature	626
5-6.	INSTRUCTION dest, source1, source2 Short Description	627
5-7.	Addressing Modes	628

5-8.	Shift Field Encoding	629
5-9.	Condition Field Encoding	629
5-10.	General Instructions	630
5-11.	Pipeline Activity For MBCNDD, Branch Not Taken	645
5-12.	Pipeline Activity For MBCNDD, Branch Taken	645
5-13.	Pipeline Activity For MCCNDD, Call Not Taken	651
5-14.	Pipeline Activity For MCCNDD, Call Taken	651
5-15.	Pipeline Activity For MMOV16 MARx, MRa , #16l	683
5-16.	Pipeline Activity For MMOV16 MAR0/MAR1, mem16	686
5-17.	Pipeline Activity For MMOVI16 MAR0/MAR1, #16l	700
5-18.	Pipeline Activity For MRCNDD, Return Not Taken	722
5-19.	Pipeline Activity For MRCNDD, Return Taken	722
5-20.	Pipeline Activity For MSTOP	726
5-21.	CLA Base Address Table	741
5-22.	CLA_REGS Registers	742
5-23.	MVECT1 Register Field Descriptions	743
5-24.	MVECT2 Register Field Descriptions	744
5-25.	MVECT3 Register Field Descriptions	745
5-26.	MVECT4 Register Field Descriptions	746
5-27.	MVECT5 Register Field Descriptions	747
5-28.	MVECT6 Register Field Descriptions	748
5-29.	MVECT7 Register Field Descriptions	749
5-30.	MVECT8 Register Field Descriptions	750
5-31.	MCTL Register Field Descriptions	751
5-32.	MIFR Register Field Descriptions	752
5-33.	MIOVF Register Field Descriptions	754
5-34.	MIFRC Register Field Descriptions	755
5-35.	MICLR Register Field Descriptions	756
5-36.	MICLROVF Register Field Descriptions	757
5-37.	MIER Register Field Descriptions	758
5-38.	MIRUN Register Field Descriptions	760
5-39.	_MPC Register Field Descriptions	761
5-40.	_MAR0 Register Field Descriptions	762
5-41.	_MAR1 Register Field Descriptions	763
5-42.	_MSTF Register Field Descriptions	764
5-43.	_MR0 Register Field Descriptions	767
5-44.	_MR1 Register Field Descriptions	768
5-45.	_MR2 Register Field Descriptions	769
5-46.	_MR3 Register Field Descriptions	770
5-47.	CLA_SOFTINT_REGS Registers	771
5-48.	SOFTINTEN Register Field Descriptions	772
5-49.	SOFTINTFRC Register Field Descriptions	773
6-1.	IPC Message RAM Read/Write Access	776
6-2.	IPC Command Registers	777
6-3.	IPC Base Addresses	780
6-4.	IPC_REGS_CPU1 Registers	781
6-5.	IPCACK Register Field Descriptions	782
6-6.	IPCSTS Register Field Descriptions	785
6-7.	IPCSET Register Field Descriptions	789

6-8.	IPCCLR Register Field Descriptions	792
6-9.	IPCFLG Register Field Descriptions	796
6-10.	IPCCOUNTERL Register Field Descriptions	799
6-11.	IPCCOUNTERH Register Field Descriptions	800
6-12.	IPCSENDCOM Register Field Descriptions	801
6-13.	IPCSENDADDR Register Field Descriptions	802
6-14.	IPCSENDDATA Register Field Descriptions	803
6-15.	IPCREMOTEREPLY Register Field Descriptions	804
6-16.	IPCRCVCOM Register Field Descriptions	805
6-17.	IPCRCVADDR Register Field Descriptions	806
6-18.	IPCRCVDATA Register Field Descriptions	807
6-19.	IPCLOCALREPLY Register Field Descriptions	808
6-20.	IPCBOOTSTS Register Field Descriptions	809
6-21.	IPCBOOTMODE Register Field Descriptions	810
6-22.	PUMPREQUEST Register Field Descriptions	811
6-23.	IPC_REGS_CPU2 Registers	812
6-24.	IPCAK Register Field Descriptions	813
6-25.	IPCSTS Register Field Descriptions	816
6-26.	IPCSET Register Field Descriptions	820
6-27.	IPCCLR Register Field Descriptions	823
6-28.	IPCFLG Register Field Descriptions	827
6-29.	IPCCOUNTERL Register Field Descriptions	830
6-30.	IPCCOUNTERH Register Field Descriptions	831
6-31.	IPCRCVCOM Register Field Descriptions	832
6-32.	IPCRCVADDR Register Field Descriptions	833
6-33.	IPCRCVDATA Register Field Descriptions	834
6-34.	IPCLOCALREPLY Register Field Descriptions	835
6-35.	IPCSENDCOM Register Field Descriptions	836
6-36.	IPCSENDADDR Register Field Descriptions	837
6-37.	IPCSENDDATA Register Field Descriptions	838
6-38.	IPCREMOTEREPLY Register Field Descriptions	839
6-39.	IPCBOOTSTS Register Field Descriptions	840
6-40.	IPCBOOTMODE Register Field Descriptions	841
6-41.	PUMPREQUEST Register Field Descriptions	842
7-1.	Sampling Period	848
7-2.	Sampling Frequency	848
7-3.	Case 1: Three-Sample Sampling Window Width	849
7-4.	Case 2: Six-Sample Sampling Window Width	849
7-5.	USB I/O Signal Muxing	850
7-6.	High-Speed SPI-Enabled GPIOs	850
7-7.	GPIO Configuration for High-Speed SPI	851
7-8.	GPIO and Peripheral Muxing	851
7-9.	Peripheral Muxing (multiple pins assigned)	852
7-10.	Pullup Resistors for 176-pin Packages	852
7-11.	Pullup Resistors for 100-pin Packages	853
7-12.	Output X-Bar Mux Configuration Table	856
7-13.	Input X-BAR Destinations	857
7-14.	GPIO Base Address Table	858
7-15.	INPUT_XBAR_REGS Registers	859

7-16.	INPUT1SELECT Register Field Descriptions	860
7-17.	INPUT2SELECT Register Field Descriptions	861
7-18.	INPUT3SELECT Register Field Descriptions	862
7-19.	INPUT4SELECT Register Field Descriptions	863
7-20.	INPUT5SELECT Register Field Descriptions	864
7-21.	INPUT6SELECT Register Field Descriptions	865
7-22.	INPUT7SELECT Register Field Descriptions	866
7-23.	INPUT8SELECT Register Field Descriptions	867
7-24.	INPUT9SELECT Register Field Descriptions	868
7-25.	INPUT10SELECT Register Field Descriptions.....	869
7-26.	INPUT11SELECT Register Field Descriptions.....	870
7-27.	INPUT12SELECT Register Field Descriptions.....	871
7-28.	INPUT13SELECT Register Field Descriptions.....	872
7-29.	INPUT14SELECT Register Field Descriptions.....	873
7-30.	INPUTSELECTLOCK Register Field Descriptions	874
7-31.	OUTPUT_XBAR_REGS Registers	878
7-32.	OUTPUT1MUX0TO15CFG Register Field Descriptions.....	879
7-33.	OUTPUT1MUX16TO31CFG Register Field Descriptions	882
7-34.	OUTPUT2MUX0TO15CFG Register Field Descriptions.....	885
7-35.	OUTPUT2MUX16TO31CFG Register Field Descriptions	888
7-36.	OUTPUT3MUX0TO15CFG Register Field Descriptions.....	891
7-37.	OUTPUT3MUX16TO31CFG Register Field Descriptions	894
7-38.	OUTPUT4MUX0TO15CFG Register Field Descriptions.....	897
7-39.	OUTPUT4MUX16TO31CFG Register Field Descriptions	900
7-40.	OUTPUT5MUX0TO15CFG Register Field Descriptions.....	903
7-41.	OUTPUT5MUX16TO31CFG Register Field Descriptions	906
7-42.	OUTPUT6MUX0TO15CFG Register Field Descriptions.....	909
7-43.	OUTPUT6MUX16TO31CFG Register Field Descriptions	912
7-44.	OUTPUT7MUX0TO15CFG Register Field Descriptions.....	915
7-45.	OUTPUT7MUX16TO31CFG Register Field Descriptions	918
7-46.	OUTPUT8MUX0TO15CFG Register Field Descriptions.....	921
7-47.	OUTPUT8MUX16TO31CFG Register Field Descriptions	924
7-48.	OUTPUT1MUXENABLE Register Field Descriptions	927
7-49.	OUTPUT2MUXENABLE Register Field Descriptions	931
7-50.	OUTPUT3MUXENABLE Register Field Descriptions	935
7-51.	OUTPUT4MUXENABLE Register Field Descriptions	939
7-52.	OUTPUT5MUXENABLE Register Field Descriptions	943
7-53.	OUTPUT6MUXENABLE Register Field Descriptions	947
7-54.	OUTPUT7MUXENABLE Register Field Descriptions	951
7-55.	OUTPUT8MUXENABLE Register Field Descriptions	955
7-56.	OUTPUTLATCH Register Field Descriptions	959
7-57.	OUTPUTLATCHCLR Register Field Descriptions.....	961
7-58.	OUTPUTLATCHFRC Register Field Descriptions.....	963
7-59.	OUTPUTLATCHENABLE Register Field Descriptions.....	965
7-60.	OUTPUTINV Register Field Descriptions.....	967
7-61.	OUTPUTLOCK Register Field Descriptions.....	969
7-62.	GPIO_CTRL_REGS Registers.....	970
7-63.	GPACTRL Register Field Descriptions	973
7-64.	GPAQSEL1 Register Field Descriptions.....	974

7-65.	GPAQSEL2 Register Field Descriptions	975
7-66.	GPAMUX1 Register Field Descriptions	976
7-67.	GPAMUX2 Register Field Descriptions	977
7-68.	GPADIR Register Field Descriptions	978
7-69.	GPAPUD Register Field Descriptions	980
7-70.	GPAINV Register Field Descriptions	982
7-71.	GPAODR Register Field Descriptions	984
7-72.	GPAGMUX1 Register Field Descriptions	986
7-73.	GPAGMUX2 Register Field Descriptions	987
7-74.	GPACSEL1 Register Field Descriptions	988
7-75.	GPACSEL2 Register Field Descriptions	989
7-76.	GPACSEL3 Register Field Descriptions	990
7-77.	GPACSEL4 Register Field Descriptions	991
7-78.	GPALOCK Register Field Descriptions	992
7-79.	GPACR Register Field Descriptions	994
7-80.	GPBCTRL Register Field Descriptions	996
7-81.	GPBQSEL1 Register Field Descriptions	997
7-82.	GPBQSEL2 Register Field Descriptions	998
7-83.	GPBMUX1 Register Field Descriptions	999
7-84.	GPBMUX2 Register Field Descriptions	1000
7-85.	GPBDIR Register Field Descriptions	1001
7-86.	GPBPUD Register Field Descriptions	1003
7-87.	GPBINV Register Field Descriptions	1005
7-88.	GPBODR Register Field Descriptions	1007
7-89.	GPBAMSEL Register Field Descriptions	1009
7-90.	GPBGMUX1 Register Field Descriptions	1011
7-91.	GPBGMUX2 Register Field Descriptions	1012
7-92.	GPBCSEL1 Register Field Descriptions	1013
7-93.	GPBCSEL2 Register Field Descriptions	1014
7-94.	GPBCSEL3 Register Field Descriptions	1015
7-95.	GPBCSEL4 Register Field Descriptions	1016
7-96.	GPBLOCK Register Field Descriptions	1017
7-97.	GPBCR Register Field Descriptions	1019
7-98.	GPCCTRL Register Field Descriptions	1021
7-99.	GPCQSEL1 Register Field Descriptions	1022
7-100.	GPCQSEL2 Register Field Descriptions	1023
7-101.	GPCMUX1 Register Field Descriptions	1024
7-102.	GPCMUX2 Register Field Descriptions	1025
7-103.	GPCDIR Register Field Descriptions	1026
7-104.	GPCPUD Register Field Descriptions	1028
7-105.	GPCINV Register Field Descriptions	1030
7-106.	GPCODR Register Field Descriptions	1032
7-107.	GPCGMUX1 Register Field Descriptions	1034
7-108.	GPCGMUX2 Register Field Descriptions	1035
7-109.	GPCCSEL1 Register Field Descriptions	1036
7-110.	GPCCSEL2 Register Field Descriptions	1037
7-111.	GPCCSEL3 Register Field Descriptions	1038
7-112.	GPCCSEL4 Register Field Descriptions	1039
7-113.	GPCLOCK Register Field Descriptions	1040

7-114. GPCCR Register Field Descriptions	1042
7-115. GPDCTRL Register Field Descriptions	1044
7-116. GPDQSEL1 Register Field Descriptions	1045
7-117. GPDQSEL2 Register Field Descriptions	1046
7-118. GPDMUX1 Register Field Descriptions	1047
7-119. GPDMUX2 Register Field Descriptions	1048
7-120. GPDDIR Register Field Descriptions	1049
7-121. GPDPUD Register Field Descriptions	1051
7-122. GPDINV Register Field Descriptions	1053
7-123. GPDODR Register Field Descriptions	1055
7-124. GPDGMUX1 Register Field Descriptions	1057
7-125. GPDGMUX2 Register Field Descriptions	1058
7-126. GPDCSEL1 Register Field Descriptions	1059
7-127. GPDCSEL2 Register Field Descriptions	1060
7-128. GPDCSEL3 Register Field Descriptions	1061
7-129. GPDCSEL4 Register Field Descriptions	1062
7-130. GPDLOCK Register Field Descriptions	1063
7-131. GPDCR Register Field Descriptions	1065
7-132. GPECTRL Register Field Descriptions	1067
7-133. GPEQSEL1 Register Field Descriptions	1068
7-134. GPEQSEL2 Register Field Descriptions	1069
7-135. GPEMUX1 Register Field Descriptions	1070
7-136. GPEMUX2 Register Field Descriptions	1071
7-137. GPEDIR Register Field Descriptions	1072
7-138. GPEPUD Register Field Descriptions	1074
7-139. GPEINV Register Field Descriptions	1076
7-140. GPEODR Register Field Descriptions	1078
7-141. GPEGMUX1 Register Field Descriptions	1080
7-142. GPEGMUX2 Register Field Descriptions	1081
7-143. GPECSEL1 Register Field Descriptions	1082
7-144. GPECSEL2 Register Field Descriptions	1083
7-145. GPECSEL3 Register Field Descriptions	1084
7-146. GPECSEL4 Register Field Descriptions	1085
7-147. GPELOCK Register Field Descriptions	1086
7-148. GPECR Register Field Descriptions	1088
7-149. GPFCTRL Register Field Descriptions	1090
7-150. GPFQSEL1 Register Field Descriptions	1091
7-151. GPFMUX1 Register Field Descriptions	1092
7-152. GPFDIR Register Field Descriptions	1093
7-153. GPFPUUD Register Field Descriptions	1095
7-154. GPFINV Register Field Descriptions	1097
7-155. GPFODR Register Field Descriptions	1099
7-156. GPFGMUX1 Register Field Descriptions	1101
7-157. GPFSEL1 Register Field Descriptions	1102
7-158. GPFSEL2 Register Field Descriptions	1103
7-159. GPFLOCK Register Field Descriptions	1104
7-160. GPFGR Register Field Descriptions	1106
7-161. GPIO_DATA_REGS Registers	1108
7-162. GPADAT Register Field Descriptions	1109

7-163. GPASET Register Field Descriptions.....	1111
7-164. GPACLEAR Register Field Descriptions	1113
7-165. GPATOGGLE Register Field Descriptions.....	1115
7-166. GPBDAT Register Field Descriptions.....	1117
7-167. GPBSET Register Field Descriptions.....	1119
7-168. GPBCLEAR Register Field Descriptions	1121
7-169. GPBTOGGLE Register Field Descriptions.....	1123
7-170. GPCDAT Register Field Descriptions	1125
7-171. GPCSET Register Field Descriptions.....	1127
7-172. GPCCLEAR Register Field Descriptions	1129
7-173. GPCTOGGLE Register Field Descriptions.....	1131
7-174. GPDDAT Register Field Descriptions	1133
7-175. GPDSET Register Field Descriptions.....	1135
7-176. GPDCLEAR Register Field Descriptions	1137
7-177. GPDTOGGLE Register Field Descriptions.....	1139
7-178. GPEDAT Register Field Descriptions.....	1141
7-179. GPESET Register Field Descriptions.....	1143
7-180. GPECLEAR Register Field Descriptions	1145
7-181. GPETOGGLE Register Field Descriptions.....	1147
7-182. GPFDAT Register Field Descriptions.....	1149
7-183. GPFSET Register Field Descriptions.....	1151
7-184. GPFCLEAR Register Field Descriptions	1153
7-185. GPFTOGGLE Register Field Descriptions.....	1155
7-186. XBAR_REGS Registers	1157
7-187. XBARFLG1 Register Field Descriptions.....	1158
7-188. XBARFLG2 Register Field Descriptions.....	1163
7-189. XBARFLG3 Register Field Descriptions.....	1167
7-190. XBARCLR1 Register Field Descriptions	1171
7-191. XBARCLR2 Register Field Descriptions	1174
7-192. XBARCLR3 Register Field Descriptions	1176
8-1. Analog Subsystem Base Address Table	1183
8-2. ANALOG_SUBSYS_REGS Registers.....	1184
8-3. INTOSC1TRIM Register Field Descriptions	1185
8-4. INTOSC2TRIM Register Field Descriptions	1186
8-5. TSNSCTL Register Field Descriptions	1187
8-6. LOCK Register Field Descriptions	1188
8-7. ANAREFTRIMA Register Field Descriptions	1189
8-8. ANAREFTRIMB Register Field Descriptions	1190
8-9. ANAREFTRIMC Register Field Descriptions	1191
8-10. ANAREFTRIMD Register Field Descriptions	1192
9-1. ADC Options and Configuration Levels	1195
9-2. Analog to 12-bit Digital Formulas	1198
9-3. Analog to 16-bit Digital Formulas	1198
9-4. 12-Bit Digital-to-Analog Formulas	1198
9-5. 16-Bit Digital-to-Analog Formulas	1198
9-6. Channel Selection of Input Pins	1201
9-7. Example Requirements for Multiple Signal Sampling	1203
9-8. Example Connections for Multiple Signal Sampling	1203
9-9. ADC Timings in 12-bit Mode (SYSCLK Cycles).....	1217

9-10.	ADC Timings in 16-bit Mode	1218
9-11.	ADC Base Address Table	1221
9-12.	ADC_REGS Registers	1222
9-13.	ADCCTL1 Register Field Descriptions	1224
9-14.	ADCCTL2 Register Field Descriptions	1225
9-15.	ADCBURSTCTL Register Field Descriptions	1226
9-16.	ADCINTFLG Register Field Descriptions	1228
9-17.	ADCINTFLGCLR Register Field Descriptions	1230
9-18.	ADCINTOVF Register Field Descriptions	1232
9-19.	ADCINTOVFCLR Register Field Descriptions	1233
9-20.	ADCINTSEL1N2 Register Field Descriptions	1234
9-21.	ADCINTSEL3N4 Register Field Descriptions	1236
9-22.	ADCSOCPRCTL Register Field Descriptions	1238
9-23.	ADCINTSOCSEL1 Register Field Descriptions	1241
9-24.	ADCINTSOCSEL2 Register Field Descriptions	1243
9-25.	ADCSOCFLG1 Register Field Descriptions	1245
9-26.	ADCSOCFRC1 Register Field Descriptions	1249
9-27.	ADCSOCOVF1 Register Field Descriptions	1254
9-28.	ADCSOCOVFCLR1 Register Field Descriptions	1257
9-29.	ADCSOC0CTL Register Field Descriptions	1260
9-30.	ADCSOC1CTL Register Field Descriptions	1263
9-31.	ADCSOC2CTL Register Field Descriptions	1266
9-32.	ADCSOC3CTL Register Field Descriptions	1269
9-33.	ADCSOC4CTL Register Field Descriptions	1272
9-34.	ADCSOC5CTL Register Field Descriptions	1275
9-35.	ADCSOC6CTL Register Field Descriptions	1278
9-36.	ADCSOC7CTL Register Field Descriptions	1281
9-37.	ADCSOC8CTL Register Field Descriptions	1284
9-38.	ADCSOC9CTL Register Field Descriptions	1287
9-39.	ADCSOC10CTL Register Field Descriptions	1290
9-40.	ADCSOC11CTL Register Field Descriptions	1293
9-41.	ADCSOC12CTL Register Field Descriptions	1296
9-42.	ADCSOC13CTL Register Field Descriptions	1299
9-43.	ADCSOC14CTL Register Field Descriptions	1302
9-44.	ADCSOC15CTL Register Field Descriptions	1305
9-45.	ADCEVTSTAT Register Field Descriptions	1308
9-46.	ADCEVTCLR Register Field Descriptions	1309
9-47.	ADCEVTSEL Register Field Descriptions	1310
9-48.	ADCEVTINTSEL Register Field Descriptions	1312
9-49.	ADCCOUNTER Register Field Descriptions	1314
9-50.	ADCREV Register Field Descriptions	1315
9-51.	ADCOFFTRIM Register Field Descriptions	1316
9-52.	ADCPPB1CONFIG Register Field Descriptions	1317
9-53.	ADCPPB1STAMP Register Field Descriptions	1319
9-54.	ADCPPB1OFFCAL Register Field Descriptions	1320
9-55.	ADCPPB1OFFREF Register Field Descriptions	1321
9-56.	ADCPPB1TRIPHI Register Field Descriptions	1322
9-57.	ADCPPB1TRIPLO Register Field Descriptions	1323
9-58.	ADCPPB2CONFIG Register Field Descriptions	1324

9-59.	ADCPPB2STAMP Register Field Descriptions	1326
9-60.	ADCPPB2OFFCAL Register Field Descriptions	1327
9-61.	ADCPPB2OFFREF Register Field Descriptions	1328
9-62.	ADCPPB2TRIPHI Register Field Descriptions	1329
9-63.	ADCPPB2TRIPLO Register Field Descriptions	1330
9-64.	ADCPPB3CONFIG Register Field Descriptions.....	1331
9-65.	ADCPPB3STAMP Register Field Descriptions.....	1333
9-66.	ADCPPB3OFFCAL Register Field Descriptions	1334
9-67.	ADCPPB3OFFREF Register Field Descriptions	1335
9-68.	ADCPPB3TRIPHI Register Field Descriptions	1336
9-69.	ADCPPB3TRIPLO Register Field Descriptions	1337
9-70.	ADCPPB4CONFIG Register Field Descriptions.....	1338
9-71.	ADCPPB4STAMP Register Field Descriptions.....	1340
9-72.	ADCPPB4OFFCAL Register Field Descriptions	1341
9-73.	ADCPPB4OFFREF Register Field Descriptions	1342
9-74.	ADCPPB4TRIPHI Register Field Descriptions	1343
9-75.	ADCPPB4TRIPLO Register Field Descriptions	1344
9-76.	ADCINLTRIM1 Register Field Descriptions.....	1345
9-77.	ADCINLTRIM2 Register Field Descriptions.....	1346
9-78.	ADCINLTRIM3 Register Field Descriptions.....	1347
9-79.	ADCINLTRIM4 Register Field Descriptions.....	1348
9-80.	ADCINLTRIM5 Register Field Descriptions.....	1349
9-81.	ADCINLTRIM6 Register Field Descriptions.....	1350
9-82.	ADC_RESULT_REGS Registers.....	1352
9-83.	ADCRESULT0 Register Field Descriptions.....	1353
9-84.	ADCRESULT1 Register Field Descriptions.....	1354
9-85.	ADCRESULT2 Register Field Descriptions.....	1355
9-86.	ADCRESULT3 Register Field Descriptions.....	1356
9-87.	ADCRESULT4 Register Field Descriptions.....	1357
9-88.	ADCRESULT5 Register Field Descriptions.....	1358
9-89.	ADCRESULT6 Register Field Descriptions.....	1359
9-90.	ADCRESULT7 Register Field Descriptions.....	1360
9-91.	ADCRESULT8 Register Field Descriptions.....	1361
9-92.	ADCRESULT9 Register Field Descriptions.....	1362
9-93.	ADCRESULT10 Register Field Descriptions	1363
9-94.	ADCRESULT11 Register Field Descriptions	1364
9-95.	ADCRESULT12 Register Field Descriptions	1365
9-96.	ADCRESULT13 Register Field Descriptions	1366
9-97.	ADCRESULT14 Register Field Descriptions	1367
9-98.	ADCRESULT15 Register Field Descriptions	1368
9-99.	ADCPPB1RESULT Register Field Descriptions	1369
9-100.	ADCPPB2RESULT Register Field Descriptions	1370
9-101.	ADCPPB3RESULT Register Field Descriptions	1371
9-102.	ADCPPB4RESULT Register Field Descriptions	1372
10-1.	DAC Base Address Table	1375
10-2.	DAC_REGS Registers	1376
10-3.	DACREV Register Field Descriptions	1377
10-4.	DACCTL Register Field Descriptions.....	1378
10-5.	DACVALA Register Field Descriptions	1379

10-6. DACVALS Register Field Descriptions	1380
10-7. DACOUTEN Register Field Descriptions	1381
10-8. DACLOCK Register Field Descriptions.....	1382
10-9. DACTRIM Register Field Descriptions	1383
11-1. CMPSS Base Address Table	1390
11-2. CMPSS_REGS Registers	1391
11-3. COMPCTL Register Field Descriptions	1392
11-4. COMPHYSCTL Register Field Descriptions	1394
11-5. COMPSTS Register Field Descriptions	1395
11-6. COMPSTSCLR Register Field Descriptions	1396
11-7. COMPDACCTL Register Field Descriptions	1397
11-8. DACHVALS Register Field Descriptions	1398
11-9. DACHVALA Register Field Descriptions	1399
11-10. RAMPMAXREFA Register Field Descriptions.....	1400
11-11. RAMPMAXREFS Register Field Descriptions.....	1401
11-12. RAMPDECVALA Register Field Descriptions	1402
11-13. RAMPDECVALS Register Field Descriptions	1403
11-14. RAMPSTS Register Field Descriptions.....	1404
11-15. DACLVALS Register Field Descriptions.....	1405
11-16. DACLVALA Register Field Descriptions.....	1406
11-17. RAMPDLYA Register Field Descriptions	1407
11-18. RAMPDLYS Register Field Descriptions	1408
11-19. CTRIPLFILCTL Register Field Descriptions	1409
11-20. CTRIPLFILCLKCTL Register Field Descriptions.....	1410
11-21. CTRIPHFILCTL Register Field Descriptions.....	1411
11-22. CTRIPHFILCLKCTL Register Field Descriptions	1412
11-23. COMPLOCK Register Field Descriptions	1413
12-1. Modulator Clock Modes	1419
12-2. Example Control Parameter Register.....	1421
12-3. Peak Data Values for Different OSR/Filter Combinations	1422
12-4. Peak Data Values for Different DOSR/Filter Combinations	1423
12-5. Number of Incorrect Samples Tabulated.....	1424
12-6. Shift Control Bit Configuration Settings	1426
12-7. General Registers	1429
12-8. Filter 1 Registers	1429
12-9. Filter 2 Registers	1429
12-10. Filter 3 Registers	1429
12-11. Filter 4 Registers	1430
12-12. SDFM Base Address Table	1430
12-13. SDFM_REGS Registers	1431
12-14. SDIFLG Register Field Descriptions	1432
12-15. SDIFLGCLR Register Field Descriptions	1434
12-16. SDCTL Register Field Descriptions.....	1435
12-17. SDMFILEN Register Field Descriptions	1436
12-18. SDSTATUS Register Field Descriptions	1437
12-19. SDCTLPARM1 Register Field Descriptions	1438
12-20. SDDFPARM1 Register Field Descriptions	1439
12-21. SDIPARM1 Register Field Descriptions.....	1440
12-22. SDCMPH1 Register Field Descriptions	1441

12-23. SDCMPL1 Register Field Descriptions	1442
12-24. SDCPARM1 Register Field Descriptions.....	1443
12-25. SDDATA1 Register Field Descriptions	1444
12-26. SDCTLPARM2 Register Field Descriptions	1445
12-27. SDDFPARM2 Register Field Descriptions	1446
12-28. SDIPARM2 Register Field Descriptions	1447
12-29. SDCMPH2 Register Field Descriptions	1448
12-30. SDCMPL2 Register Field Descriptions	1449
12-31. SDCPARM2 Register Field Descriptions.....	1450
12-32. SDDATA2 Register Field Descriptions	1451
12-33. SDCTLPARM3 Register Field Descriptions	1452
12-34. SDDFPARM3 Register Field Descriptions	1453
12-35. SDIPARM3 Register Field Descriptions	1454
12-36. SDCMPH3 Register Field Descriptions	1455
12-37. SDCMPL3 Register Field Descriptions	1456
12-38. SDCPARM3 Register Field Descriptions.....	1457
12-39. SDDATA3 Register Field Descriptions	1458
12-40. SDCTLPARM4 Register Field Descriptions	1459
12-41. SDDFPARM4 Register Field Descriptions	1460
12-42. SDIPARM4 Register Field Descriptions	1461
12-43. SDCMPH4 Register Field Descriptions	1462
12-44. SDCMPL4 Register Field Descriptions	1463
12-45. SDCPARM4 Register Field Descriptions.....	1464
12-46. SDDATA4 Register Field Descriptions	1465
13-1. Submodule Configuration Parameters.....	1473
13-2. Key Time-Base Signals.....	1476
13-3. Action-Qualifier Submodule Possible Input Events	1494
13-4. Action-Qualifier Event Priority for Up-Down-Count Mode.....	1496
13-5. Action-Qualifier Event Priority for Up-Count Mode.....	1496
13-6. Action-Qualifier Event Priority for Down-Count Mode	1496
13-7. Behavior if CMPA/CMPB is Greater than the Period	1497
13-8. Classical Dead-Band Operating Modes	1509
13-9. Additional Dead-Band Operating Modes	1510
13-10. Dead-Band Delay Values in μ S as a Function of DBFED and DBRED	1512
13-11. Possible Pulse Width Values for EPWMCLK = 80 MHz	1515
13-12. Possible Actions On a Trip Event	1519
13-13. ePWM X-BAR Mux Configuration Table	1538
13-14. EPWM Base Address Table	1557
13-15. EPWM_REGS Registers	1558
13-16. TBCTL Register Field Descriptions	1560
13-17. TBCTL2 Register Field Descriptions.....	1562
13-18. TBCTR Register Field Descriptions.....	1563
13-19. TBSTS Register Field Descriptions	1564
13-20. CMPCTL Register Field Descriptions.....	1565
13-21. CMPCTL2 Register Field Descriptions	1567
13-22. DBCTL Register Field Descriptions	1569
13-23. DBCTL2 Register Field Descriptions	1572
13-24. AQCTL Register Field Descriptions.....	1573
13-25. AQTSRCSEL Register Field Descriptions	1575

13-26. PCCTL Register Field Descriptions	1576
13-27. HRCNFG Register Field Descriptions	1578
13-28. HRPWR Register Field Descriptions	1580
13-29. HRMSTEP Register Field Descriptions	1581
13-30. HRCNFG2 Register Field Descriptions.....	1582
13-31. HRPCTL Register Field Descriptions.....	1583
13-32. GLDCTL Register Field Descriptions	1585
13-33. GLDCFG Register Field Descriptions	1587
13-34. EPWMXLINK Register Field Descriptions	1589
13-35. AQCTLA Register Field Descriptions.....	1593
13-36. AQCTLA2 Register Field Descriptions	1595
13-37. AQCTLB Register Field Descriptions.....	1596
13-38. AQCTLB2 Register Field Descriptions	1598
13-39. AQSFRD Register Field Descriptions	1599
13-40. AQCSFRD Register Field Descriptions	1600
13-41. DBREDHR Register Field Descriptions	1601
13-42. DBRED Register Field Descriptions	1602
13-43. DBFEDHR Register Field Descriptions.....	1603
13-44. DBFED Register Field Descriptions	1604
13-45. TBPHS Register Field Descriptions.....	1605
13-46. TBPRDHR Register Field Descriptions.....	1606
13-47. TBPRD Register Field Descriptions	1607
13-48. CMPA Register Field Descriptions	1608
13-49. CMPB Register Field Descriptions	1609
13-50. CMPC Register Field Descriptions.....	1610
13-51. CMPD Register Field Descriptions.....	1611
13-52. GLDCTL2 Register Field Descriptions	1612
13-53. TZSEL Register Field Descriptions	1613
13-54. TZDSEL Register Field Descriptions	1615
13-55. TZCTL Register Field Descriptions	1617
13-56. TZCTL2 Register Field Descriptions.....	1619
13-57. TZCTLDCA Register Field Descriptions.....	1621
13-58. TZCTLCB Register Field Descriptions.....	1623
13-59. TZEINT Register Field Descriptions	1625
13-60. TZFLG Register Field Descriptions	1626
13-61. TZCBCFLG Register Field Descriptions	1628
13-62. TZOSTFLG Register Field Descriptions.....	1629
13-63. TZCLR Register Field Descriptions	1630
13-64. TZCBCCLR Register Field Descriptions	1631
13-65. TZOSTCLR Register Field Descriptions	1632
13-66. TZFRC Register Field Descriptions.....	1633
13-67. ETSEL Register Field Descriptions	1634
13-68. ETPS Register Field Descriptions.....	1637
13-69. ETFLG Register Field Descriptions	1640
13-70. ETCLR Register Field Descriptions.....	1641
13-71. ETFRC Register Field Descriptions.....	1642
13-72. ETINTPS Register Field Descriptions	1643
13-73. ETSOCPS Register Field Descriptions.....	1644
13-74. ETCNTINITCTL Register Field Descriptions	1646

13-75. ETCNTINIT Register Field Descriptions.....	1647
13-76. DCTRISEL Register Field Descriptions	1648
13-77. DCACTL Register Field Descriptions.....	1650
13-78. DCBCTL Register Field Descriptions	1651
13-79. DCFCTL Register Field Descriptions	1652
13-80. DCCAPCTL Register Field Descriptions	1653
13-81. DCFOFFSET Register Field Descriptions	1654
13-82. DCFOFFSETCNT Register Field Descriptions.....	1655
13-83. DCFWINDOW Register Field Descriptions	1656
13-84. DCFWINDOWCNT Register Field Descriptions.....	1657
13-85. DCCAP Register Field Descriptions	1658
13-86. DCAHTRIPSEL Register Field Descriptions.....	1659
13-87. DCALTRIPSEL Register Field Descriptions	1661
13-88. DCBHTRIPSEL Register Field Descriptions.....	1663
13-89. DCBLTRIPSEL Register Field Descriptions	1665
13-90. EPWM_XBAR_REGS Registers	1667
13-91. TRIP4MUX0TO15CFG Register Field Descriptions	1668
13-92. TRIP4MUX16TO31CFG Register Field Descriptions.....	1671
13-93. TRIP5MUX0TO15CFG Register Field Descriptions	1674
13-94. TRIP5MUX16TO31CFG Register Field Descriptions.....	1677
13-95. TRIP7MUX0TO15CFG Register Field Descriptions	1680
13-96. TRIP7MUX16TO31CFG Register Field Descriptions.....	1683
13-97. TRIP8MUX0TO15CFG Register Field Descriptions	1686
13-98. TRIP8MUX16TO31CFG Register Field Descriptions.....	1689
13-99. TRIP9MUX0TO15CFG Register Field Descriptions	1692
13-100. TRIP9MUX16TO31CFG Register Field Descriptions	1695
13-101. TRIP10MUX0TO15CFG Register Field Descriptions	1698
13-102. TRIP10MUX16TO31CFG Register Field Descriptions.....	1701
13-103. TRIP11MUX0TO15CFG Register Field Descriptions	1704
13-104. TRIP11MUX16TO31CFG Register Field Descriptions.....	1707
13-105. TRIP12MUX0TO15CFG Register Field Descriptions	1710
13-106. TRIP12MUX16TO31CFG Register Field Descriptions.....	1713
13-107. TRIP4MUXENABLE Register Field Descriptions	1716
13-108. TRIP5MUXENABLE Register Field Descriptions	1720
13-109. TRIP7MUXENABLE Register Field Descriptions	1724
13-110. TRIP8MUXENABLE Register Field Descriptions	1728
13-111. TRIP9MUXENABLE Register Field Descriptions	1732
13-112. TRIP10MUXENABLE Register Field Descriptions	1736
13-113. TRIP11MUXENABLE Register Field Descriptions	1740
13-114. TRIP12MUXENABLE Register Field Descriptions	1744
13-115. TRIPOUTINV Register Field Descriptions.....	1748
13-116. TRIPLOCK Register Field Descriptions.....	1750
13-117. TRIG_REGS Registers.....	1751
13-118. SYNCSELECT Register Field Descriptions	1752
13-119. EXTADCSOCSELECT Register Field Descriptions	1754
13-120. SYNCSOCLOCK Register Field Descriptions.....	1757
14-1. Resolution for PWM and HRPWM	1760
14-2. Relationship Between MEP Steps, PWM Frequency and Resolution.....	1766
14-3. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right).....	1767

14-4. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles	1770
14-5. SFO Library Features.....	1782
14-6. Factor Values	1783
15-1. eCAP Base Address Table	1807
15-2. ECAP_REGS Registers	1808
15-3. TSCTR Register Field Descriptions.....	1809
15-4. CTRPHS Register Field Descriptions.....	1810
15-5. CAP1 Register Field Descriptions.....	1811
15-6. CAP2 Register Field Descriptions.....	1812
15-7. CAP3 Register Field Descriptions.....	1813
15-8. CAP4 Register Field Descriptions.....	1814
15-9. ECCTL1 Register Field Descriptions	1815
15-10. ECCTL2 Register Field Descriptions	1817
15-11. ECEINT Register Field Descriptions.....	1819
15-12. ECFLG Register Field Descriptions.....	1820
15-13. ECCLR Register Field Descriptions	1821
15-14. ECFRC Register Field Descriptions	1822
16-1. EQEP Memory Map	1828
16-2. Quadrature Decoder Truth Table	1830
16-3. eQEP Base Address Table	1843
16-4. EQEP_REGS Registers	1844
16-5. QPOSCNT Register Field Descriptions	1845
16-6. QPOSINIT Register Field Descriptions.....	1846
16-7. QPOSMAX Register Field Descriptions	1847
16-8. QPOSCMP Register Field Descriptions	1848
16-9. QPOSILAT Register Field Descriptions	1849
16-10. QPOSSLAT Register Field Descriptions	1850
16-11. QPOSLAT Register Field Descriptions	1851
16-12. QUTMR Register Field Descriptions.....	1852
16-13. QUPRD Register Field Descriptions	1853
16-14. QWDTMR Register Field Descriptions	1854
16-15. QWDPRD Register Field Descriptions	1855
16-16. QDECCTL Register Field Descriptions.....	1856
16-17. QEPCTL Register Field Descriptions.....	1857
16-18. QCAPCTL Register Field Descriptions.....	1860
16-19. QPOSCTL Register Field Descriptions.....	1861
16-20. QEINT Register Field Descriptions	1862
16-21. QFLG Register Field Descriptions	1863
16-22. QCLR Register Field Descriptions	1865
16-23. QFRC Register Field Descriptions	1867
16-24. QEPSTS Register Field Descriptions.....	1868
16-25. QCTMR Register Field Descriptions.....	1869
16-26. QCPRD Register Field Descriptions.....	1870
16-27. QCTMRLAT Register Field Descriptions.....	1871
16-28. QCPRDLAT Register Field Descriptions	1872
17-1. SPI Module Signal Summary	1875
17-2. SPI Clocking Scheme Selection Guide.....	1882
17-3. SPI Interrupt Flag Modes.....	1886
17-4. High-Speed SPI Capable GPIOs.....	1888

17-5. 4-wire vs. 3-wire SPI Pin Functions	1889
17-6. 3-Wire SPI Pin Configuration	1890
17-7. SPI Base Address Table	1898
17-8. SPI_REGS Registers	1899
17-9. SPICCR Register Field Descriptions	1900
17-10. SPICTL Register Field Descriptions	1902
17-11. SPISTS Register Field Descriptions	1904
17-12. SPIBRR Register Field Descriptions	1906
17-13. SPIRXEMU Register Field Descriptions	1907
17-14. SPIRXBUF Register Field Descriptions	1908
17-15. SPITXBUF Register Field Descriptions	1909
17-16. SPIDAT Register Field Descriptions	1910
17-17. SPIFFTX Register Field Descriptions	1911
17-18. SPIFFRX Register Field Descriptions	1913
17-19. SPIFFCT Register Field Descriptions	1915
17-20. SPIPRI Register Field Descriptions	1916
18-1. SCI Module Signal Summary	1920
18-2. Programming the Data Format Using SCICCR	1921
18-3. Asynchronous Baud Register Values for Common SCI Bit Rates	1927
18-4. SCI Interrupt Flags	1929
18-5. SCI Base Address Table	1931
18-6. SCI_REGS Registers	1932
18-7. SCICCR Register Field Descriptions	1933
18-8. SCICTL1 Register Field Descriptions	1935
18-9. SCIHBAUD Register Field Descriptions	1937
18-10. SCILBAUD Register Field Descriptions	1938
18-11. SCICTL2 Register Field Descriptions	1939
18-12. SCIRXST Register Field Descriptions	1940
18-13. SCIRXEMU Register Field Descriptions	1942
18-14. SCIRXBUF Register Field Descriptions	1943
18-15. SCITXBUF Register Field Descriptions	1944
18-16. SCIFFTX Register Field Descriptions	1945
18-17. SCIFFRX Register Field Descriptions	1947
18-18. SCIFFCT Register Field Descriptions	1949
18-19. SCIPRI Register Field Descriptions	1950
19-1. Dependency of Delay d on the Divide-Down Value IPSC	1956
19-2. Operating Modes of the I2C Module	1957
19-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR	1957
19-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR	1959
19-5. Ways to Generate a NACK Bit	1960
19-6. Descriptions of the Basic I2C Interrupt Requests	1963
19-7. I2C Base Address Table	1966
19-8. I2C_REGS Registers	1967
19-9. I2COAR Register Field Descriptions	1968
19-10. I2CIER Register Field Descriptions	1969
19-11. I2CSTR Register Field Descriptions	1970
19-12. I2CCLKL Register Field Descriptions	1974
19-13. I2CCLKH Register Field Descriptions	1975
19-14. I2CCNT Register Field Descriptions	1976

19-15. I2CDRR Register Field Descriptions	1977
19-16. I2CSAR Register Field Descriptions	1978
19-17. I2CDXR Register Field Descriptions	1979
19-18. I2CMDR Register Field Descriptions	1980
19-19. I2CISRC Register Field Descriptions	1984
19-20. I2CEMDR Register Field Descriptions	1985
19-21. I2CPSC Register Field Descriptions	1986
19-22. I2CFFTX Register Field Descriptions	1987
19-23. I2CFFRX Register Field Descriptions	1988
20-1. McBSP Interface Pins/Signals	1991
20-2. Register Bits That Determine the Number of Phases, Words, and Bits	1998
20-3. Interrupts and DMA Events Generated by a McBSP	2002
20-4. Effects of DLB and CLKSTP on Clock Modes.....	2004
20-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits	2004
20-6. Polarity Options for the Input to the Sample Rate Generator	2005
20-7. Input Clock Selection for Sample Rate Generator	2008
20-8. Block - Channel Assignment.....	2017
20-9. 2-Partition Mode	2018
20-10. 8-Partition mode	2018
20-11. Receive Channel Assignment and Control With Eight Receive Partitions.....	2020
20-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used	2021
20-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits.....	2022
20-14. Bits Used to Enable and Configure the Clock Stop Mode	2025
20-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme	2026
20-16. Bit Values Required to Configure the McBSP as an SPI Master	2029
20-17. Bit Values Required to Configure the McBSP as an SPI Slave.....	2030
20-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions	2032
20-19. Reset State of Each McBSP Pin	2032
20-20. Register Bit Used to Enable/Disable the Digital Loopback Mode.....	2033
20-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode	2033
20-22. Register Bits Used to Enable/Disable the Clock Stop Mode	2033
20-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme	2034
20-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode	2034
20-25. Register Bit Used to Choose One or Two Phases for the Receive Frame	2034
20-26. Register Bits Used to Set the Receive Word Length(s).....	2035
20-27. Register Bits Used to Set the Receive Frame Length.....	2035
20-28. How to Calculate the Length of the Receive Frame	2036
20-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function	2036
20-30. Register Bits Used to Set the Receive Companding Mode.....	2037
20-31. Register Bits Used to Set the Receive Data Delay	2038
20-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode.....	2040
20-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh.....	2040
20-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh	2040
20-35. Register Bits Used to Set the Receive Interrupt Mode	2041
20-36. Register Bits Used to Set the Receive Frame Synchronization Mode	2041
20-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin.....	2042
20-38. Register Bit Used to Set Receive Frame-Synchronization Polarity.....	2043
20-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width	2044
20-40. Register Bits Used to Set the Receive Clock Mode	2045

20-41. Receive Clock Signal Source Selection	2046
20-42. Register Bit Used to Set Receive Clock Polarity	2046
20-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value	2048
20-44. Register Bit Used to Set the SRG Clock Synchronization Mode	2048
20-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)	2049
20-46. Register Bits Used to Set the SRG Input Clock Polarity	2050
20-47. Register Bits Used to Place Transmitter in Reset Field Descriptions	2051
20-48. Register Bit Used to Enable/Disable the Digital Loopback Mode	2052
20-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode	2052
20-50. Register Bits Used to Enable/Disable the Clock Stop Mode	2052
20-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme	2053
20-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection	2054
20-53. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame	2055
20-54. Register Bits Used to Set the Transmit Word Length(s)	2055
20-55. Register Bits Used to Set the Transmit Frame Length	2056
20-56. How to Calculate Frame Length	2056
20-57. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function	2057
20-58. Register Bits Used to Set the Transmit Companding Mode	2058
20-59. Register Bits Used to Set the Transmit Data Delay	2059
20-60. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode	2061
20-61. Register Bits Used to Set the Transmit Interrupt Mode	2061
20-62. Register Bits Used to Set the Transmit Frame-Synchronization Mode	2062
20-63. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses	2062
20-64. Register Bit Used to Set Transmit Frame-Synchronization Polarity	2063
20-65. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width	2064
20-66. Register Bit Used to Set the Transmit Clock Mode	2065
20-67. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin	2065
20-68. Register Bit Used to Set Transmit Clock Polarity	2065
20-69. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2	2067
20-70. Reset State of Each McBSP Pin	2067
20-71. Receive Interrupt Sources and Signals	2072
20-72. Transmit Interrupt Sources and Signals	2072
20-73. Error Flags	2073
20-74. McBSP Interrupt Enable Register (MFFINT) Field Descriptions	2073
20-75. McBSP Mode Selection	2074
20-76. McBSP Base Address Table	2075
20-77. McBSP Register Summary	2075
20-78. Serial Port Control 1 Register (SPCR1) Field Descriptions	2077
20-79. Serial Port Control 2 Register (SPCR2) Field Descriptions	2080
20-80. Receive Control Register 1 (RCR1) Field Descriptions	2082
20-81. Frame Length Formula for Receive Control 1 Register (RCR1)	2083
20-82. Receive Control Register 2 (RCR2) Field Descriptions	2083
20-83. Frame Length Formula for Receive Control 2 Register (RCR2)	2084
20-84. Transmit Control 1 Register (XCR1) Field Descriptions	2085
20-85. Frame Length Formula for Transmit Control 1 Register (XCR1)	2085
20-86. Transmit Control 2 Register (XCR2) Field Descriptions	2086
20-87. Frame Length Formula for Transmit Control 2 Register (XCR2)	2087
20-88. Sample Rate Generator 1 Register (SRGR1) Field Descriptions	2088
20-89. Sample Rate Generator 2 Register (SRGR2) Field Descriptions	2089

20-90. Multichannel Control 1 Register (MCR1) Field Descriptions	2090
20-91. Multichannel Control 2 Register (MCR2) Field Descriptions	2092
20-92. Pin Control Register (PCR) Field Descriptions	2094
20-93. Pin Configuration	2096
20-94. Receive Channel Enable Registers (RCERA...RCERH) Field Descriptions.....	2096
20-95. Use of the Receive Channel Enable Registers	2097
20-96. Transmit Channel Enable Registers (XCERA...XCERH) Field Descriptions	2098
20-97. Use of the Transmit Channel Enable Registers	2099
21-1. Programmable Ranges Required by CAN Protocol	2119
21-2. Message Object Field Descriptions	2128
21-3. Message RAM Addressing in Debug Mode	2131
21-4. CAN Base Addresses Table	2133
21-5. CAN_REGS Registers.....	2134
21-6. CAN_CTL Register Field Descriptions	2135
21-7. CAN_ES Register Field Descriptions.....	2137
21-8. CAN_ERRC Register Field Descriptions.....	2139
21-9. CAN_BTR Register Field Descriptions	2140
21-10. CAN_INT Register Field Descriptions	2141
21-11. CAN_TEST Register Field Descriptions.....	2142
21-12. CAN_PERR Register Field Descriptions	2143
21-13. CAN_REL Register Field Descriptions	2144
21-14. CAN_RAM_INIT Register Field Descriptions.....	2145
21-15. CAN_GLB_INT_EN Register Field Descriptions	2146
21-16. CAN_GLB_INT_FLG Register Field Descriptions	2147
21-17. CAN_GLB_INT_CLR Register Field Descriptions	2148
21-18. CAN_ABOTR Register Field Descriptions	2149
21-19. CAN_TXRQ_X Register Field Descriptions.....	2150
21-20. CAN_TXRQ_21 Register Field Descriptions	2151
21-21. CAN_NDAT_X Register Field Descriptions.....	2152
21-22. CAN_NDAT_21 Register Field Descriptions.....	2153
21-23. CAN_IPEN_X Register Field Descriptions	2154
21-24. CAN_IPEN_21 Register Field Descriptions.....	2155
21-25. CAN_MVAL_X Register Field Descriptions.....	2156
21-26. CAN_MVAL_21 Register Field Descriptions.....	2157
21-27. CAN_IP_MUX21 Register Field Descriptions	2158
21-28. CAN_IF1CMD Register Field Descriptions	2159
21-29. CAN_IF1MSK Register Field Descriptions.....	2162
21-30. CAN_IF1ARB Register Field Descriptions	2163
21-31. CAN_IF1MCTL Register Field Descriptions	2164
21-32. CAN_IF1DATA Register Field Descriptions	2166
21-33. CAN_IF1DATB Register Field Descriptions	2167
21-34. CAN_IF2CMD Register Field Descriptions	2168
21-35. CAN_IF2MSK Register Field Descriptions.....	2171
21-36. CAN_IF2ARB Register Field Descriptions	2172
21-37. CAN_IF2MCTL Register Field Descriptions	2173
21-38. CAN_IF2DATA Register Field Descriptions	2175
21-39. CAN_IF2DATB Register Field Descriptions	2176
21-40. CAN_IF3OBS Register Field Descriptions	2177
21-41. CAN_IF3MSK Register Field Descriptions.....	2179

21-42. CAN_IF3ARB Register Field Descriptions	2180
21-43. CAN_IF3MCTL Register Field Descriptions	2181
21-44. CAN_IF3DATA Register Field Descriptions	2183
21-45. CAN_IF3DATB Register Field Descriptions	2184
21-46. CAN_IF3UPD Register Field Descriptions	2185
22-1. USB Memory Access From Software.....	2196
22-2. USB Memory Access From CCS.....	2197
22-3. Universal Serial Bus (USB) Controller Register Map	2199
22-4. Function Address Register (USBFADDR) Field Descriptions	2204
22-5. Power Management Register (USBPOWER) in Host Mode Field Descriptions	2205
22-6. Power Management Register (USBPOWER) in Device Mode Field Descriptions.....	2205
22-7. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions	2207
22-8. USB Receive Interrupt Status Register (USBRXIS) Field Descriptions.....	2208
22-9. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions	2209
22-10. USB Receive Interrupt Register (USBRXIE) Field Descriptions	2210
22-11. USB General Interrupt Status Register (USBIS) in Host Mode Field Descriptions.....	2211
22-12. USB General Interrupt Status Register (USBIS) in Device Mode Field Descriptions	2212
22-13. USB Interrupt Enable Register (USBIE) in Host Mode Field Descriptions	2213
22-14. USB Interrupt Enable Register (USBIE) in Device Mode Field Descriptions.....	2214
22-15. Frame Number Register (FRAME) Field Descriptions	2215
22-16. USB Endpoint Index Register (USBEPIDX) Field Descriptions	2215
22-17. USB Test Mode Register (USBTEST) in Host Mode Field Descriptions.....	2216
22-18. USB Test Mode Register (USBTEST) in Device Mode Field Descriptions.....	2216
22-19. USB FIFO Endpoint <i>n</i> Register (USBFIFO[<i>n</i>]) Field Descriptions	2218
22-20. USB Device Control Register (USBDEVCTL) Field Descriptions.....	2219
22-21. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions	2221
22-22. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions.....	2222
22-23. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions	2223
22-24. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions.....	2224
22-25. USB Connect Timing Register (USBCONTIM) Field Descriptions.....	2225
22-26. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions	2226
22-27. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions.....	2226
22-28. USB Transmit Functional Address Endpoint <i>n</i> Registers (USBTXFUNCADDR[<i>n</i>]) Field Descriptions	2227
22-29. USB Transmit Hub Address Endpoint <i>n</i> Registers(USBTXHUBADDR[<i>n</i>]) Field Descriptions	2228
22-30. USB Transmit Hub Port Endpoint <i>n</i> Registers(USBTXHUBPORT[<i>n</i>]) Field Descriptions	2229
22-31. USB Recieve Functional Address Endpoint <i>n</i> Registers(USBFIFO[<i>n</i>]) Field Descriptions	2230
22-32. USB Receive Hub Address Endpoint <i>n</i> Registers(USBRXHUBADDR[<i>n</i>]) Field Descriptions	2231
22-33. USB Transmit Hub Port Endpoint <i>n</i> Registers(USBRXHUBPORT[<i>n</i>]) Field Descriptions	2232
22-34. USB Maximum Transmit Data Endpoint <i>n</i> Registers(USBTXMAXP[<i>n</i>]) Field Descriptions	2233
22-35. USB Control and Status Endpoint 0 Low Register(USBCSRL0) in Host Mode Field Descriptions	2234
22-36. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode Field Descriptions	2235
22-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode Field Descriptions.....	2236
22-38. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode Field Descriptions.....	2236
22-39. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions	2237
22-40. USB Type Endpoint 0 Register (USBTYP0) Field Descriptions.....	2237
22-41. USB NAK Limit Register (USBNAKLMT) Field Descriptions	2238
22-42. USB Transmit Control and Status Endpoint <i>n</i> Low Register (USBTXCSRL[<i>n</i>]) in Host Mode Field Descriptions	2239
22-43. USB Transmit Control and Status Endpoint <i>n</i> Low Register (USBTXCSRL[<i>n</i>]) in Device Mode Field	

Descriptions	2240
22-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n]) in Host Mode Field Descriptions	2242
22-45. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n]) in Device Mode Field Descriptions	2243
22-46. USB Maximum Receive Data Endpoint n Registers (USBTXMAXP[n]) Field Descriptions	2244
22-47. USB Control and Status Endpoint n Low Register(USBCSRL[n]) in Host Mode Field Descriptions	2245
22-48. USB Control and Status Endpoint 0 Low Register(USBCSRL[n]) in Device Mode Field Descriptions	2246
22-49. USB Control and Status Endpoint n High Register (USBCSRH[n]) in Host Mode Field Descriptions	2248
22-50. USB Control and Status Endpoint 0 High Register(USBCSRH[n]) in Device Mode Field Descriptions	2249
22-51. USB Maximum Receive Data Endpoint n Registers (USBRXCOUNT[n]) Field Descriptions	2250
22-52. USB Host Transmit Configure Type Endpoint n Register(USBTXTYPE[n]) Field Descriptions	2251
22-53. USBTXINTERVAL[n] Frame Numbers	2252
22-54. USB Host Transmit Interval Endpoint n Register(USBTXINTERVAL[n]) Field Descriptions	2252
22-55. USB Host Configure Receive Type Endpoint n Register(USBRXTYPE[n]) Field Descriptions	2253
22-56. USBRXINTERVAL[n] Frame Numbers	2254
22-57. USB Host Receive Polling Interval Endpoint n Register(USBRXINTERVAL[n]) Field Descriptions.....	2254
22-58. USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[n]) Field Descriptions	2255
22-59. USB Receive Double Packet Buffer Disable Register (USBRXDPKTBUFDIS) Field Descriptions	2256
22-60. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions	2257
22-61. USB External Power Control Register (USBEPD) Field Descriptions.....	2258
22-62. USB External Power Control Raw Interrupt Status Register (USBEPDRIS) Field Descriptions.....	2260
22-63. USB External Power Control Interrupt Mask Register (USBEPDIM) Field Descriptions.....	2261
22-64. USB External Power Control Interrupt Status and Clear Register (USBEPDISC) Field Descriptions	2262
22-65. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions.....	2263
22-66. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions.....	2264
22-67. USB Device RESUME Interrupt Status and Clear Register (USBDRISC) Field Descriptions	2265
22-68. USB General-Purpose Control and Status Register (USBGPCS) Field Descriptions.....	2266
22-69. USB DMA Select Register (USBDMASEL) Field Descriptions.....	2267
23-1. uPP Signal Description	2274
23-2. CPU/CLA/uPP-DMA Address Map.....	2281
23-3. CPU/CLA/uPP-DMA Address Map.....	2282
23-4. uPP Parameters Useful for System Tuning.....	2283
23-5. UPP Base Address Table	2284
23-6. UPP_REGS Registers	2285
23-7. PID Register Field Descriptions	2286
23-8. PERCTL Register Field Descriptions	2287
23-9. CHCTL Register Field Descriptions.....	2288
23-10. IFCFG Register Field Descriptions.....	2289
23-11. IFIVAL Register Field Descriptions	2291
23-12. THCFG Register Field Descriptions	2292
23-13. RAWINTST Register Field Descriptions.....	2294
23-14. ENINTST Register Field Descriptions	2296
23-15. INTENSET Register Field Descriptions	2298
23-16. INTENCLR Register Field Descriptions	2300
23-17. CHIDESC0 Register Field Descriptions	2302
23-18. CHIDESC1 Register Field Descriptions	2303
23-19. CHIDESC2 Register Field Descriptions	2304
23-20. CHIST0 Register Field Descriptions	2305

23-21. CHIST1 Register Field Descriptions	2306
23-22. CHIST2 Register Field Descriptions	2307
23-23. CHQDESC0 Register Field Descriptions.....	2308
23-24. CHQDESC1 Register Field Descriptions.....	2309
23-25. CHQDESC2 Register Field Descriptions.....	2310
23-26. CHQST0 Register Field Descriptions.....	2311
23-27. CHQST1 Register Field Descriptions.....	2312
23-28. CHQST2 Register Field Descriptions.....	2313
23-29. GINTEN Register Field Descriptions	2314
23-30. GINTFLG Register Field Descriptions	2315
23-31. GINTCLR Register Field Descriptions	2316
23-32. DLYCTL Register Field Descriptions	2317
24-1. Configuration for EMIF1 and EMIF2 Modules	2319
24-2. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories	2322
24-3. EMIF Pins Specific to SDRAM	2322
24-4. EMIF Pins Specific to Asynchronous Memory	2323
24-5. EMIF SDRAM Commands	2323
24-6. Truth Table for SDRAM Commands	2324
24-7. 16-bit EMIF Address Pin Connections.....	2326
24-8. Description of the SDRAM Configuration Register (SDRAM_CR).....	2327
24-9. Description of the SDRAM Refresh Control Register (SDRAM_RCR).....	2327
24-10. Description of the SDRAM Timing Register (SDRAM_TR)	2328
24-11. Description of the SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)	2328
24-12. SDRAM LOAD MODE REGISTER Command	2329
24-13. Refresh Urgency Levels	2330
24-14. Mapping from Logical Address to EMIF Pins for 32-bit SDRAM.....	2335
24-15. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM.....	2335
24-16. Normal Mode vs. Select Strobe Mode.....	2336
24-17. Description of the Asynchronous <i>m</i> Configuration Register (ASYNC_CS _n _CR)	2338
24-18. Description of the Asynchronous Wait Cycle Configuration Register (ASYNC_WCCR)	2339
24-19. Description of EMIF Interrupt Mask Set Register (INT_MSK_SET)	2340
24-20. Description of EMIF Interrupt Mast Clear Register (INT_MSK_CLR)	2340
24-21. Asynchronous Read Operation in Normal Mode	2340
24-22. Asynchronous Write Operation in Normal Mode	2342
24-23. Asynchronous Read Operation in Select Strobe Mode	2344
24-24. Asynchronous Write Operation in Select Strobe Mode	2346
24-25. Interrupt Monitor and Control Bit Fields	2350
24-26. SR Field Value For EMIF to K4S641632H-TC(L)70 Interface.....	2354
24-27. SDRAM_TR Field Calculations for EMIF to K4S641632H-TC(L)70 Interface.....	2356
24-28. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface	2357
24-29. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface	2357
24-30. SDRAM_CR Field Values For EMIF to K4S641632H-TC(L)70 Interface	2358
24-31. AC Characteristics for a Read Access	2359
24-32. AC Characteristics for a Write Access	2359
24-33. EMIF Base Address Table	2362
24-34. EMIF_REGS Registers	2363
24-35. RCSR Register Field Descriptions	2364
24-36. ASYNC_WCCR Register Field Descriptions	2365
24-37. SDRAM_CR Register Field Descriptions	2366

24-38. SDRAM_RCR Register Field Descriptions	2368
24-39. ASYNC_CS2_CR Register Field Descriptions	2369
24-40. ASYNC_CS3_CR Register Field Descriptions	2370
24-41. ASYNC_CS4_CR Register Field Descriptions	2371
24-42. SDRAM_TR Register Field Descriptions	2372
24-43. TOTAL_SDRAM_AR Register Field Descriptions	2373
24-44. TOTAL_SDRAM_ACTR Register Field Descriptions	2374
24-45. SDR_EXT_TMNG Register Field Descriptions	2375
24-46. INT_RAW Register Field Descriptions	2376
24-47. INT_MSK Register Field Descriptions	2377
24-48. INT_MSK_SET Register Field Descriptions	2378
24-49. INT_MSK_CLR Register Field Descriptions	2379
24-50. EMIF1_CONFIG_REGS Registers	2380
24-51. EMIF1LOCK Register Field Descriptions	2381
24-52. EMIF1COMMIT Register Field Descriptions	2382
24-53. EMIF1MSEL Register Field Descriptions	2383
24-54. EMIF1ACCPROT0 Register Field Descriptions	2384
24-55. EMIF2_CONFIG_REGS Registers	2385
24-56. EMIF2LOCK Register Field Descriptions	2386
24-57. EMIF2COMMIT Register Field Descriptions	2387
24-58. EMIF2ACCPROT0 Register Field Descriptions	2388

Read This First

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 F2837xD Microcontroller processors.

This TRM has been designated *Preliminary* because the documentation is in the formative or design phase of development. Texas Instruments reserves the right to change this TRM without notice. Visit the Texas Instruments website at <http://www.ti.com> to determine the latest version of this TRM.

Notational Conventions

These documents use the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

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.

Trademarks

Code Composer Studio, Delfino are trademarks of Texas Instruments. USB Specification Revision 2.0 is a trademark of Compaq Computer Corp. XDS is a trademark of XDS.

C28x Processor

This chapter contains a modified description of the C28x Processor and provides links to access their respective references guides.

Topic	Page
1.1 Overview	80

1.1 Overview

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

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. The CPU does this over six separate address/data buses.

For more information on CPU architecture and instruction set, see the *TMS320C28x CPU and Instruction Set Reference Guide* (literature number [SPRU430](#)). For more information on the C28x Floating Point Unit (FPU), Trigonometric Math Unit, and Viterbi, Complex Math, and CRC Unit II (VCU-II) instruction sets, see the *TMS320C28x Extended Instruction Sets Technical Reference Guide* (literature number [SPRUHS1](#)). A brief overview of the FPU, TMU, and VCU-II are provided here.

1.1.1 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.

1.1.2 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 1-1](#).

Table 1-1. 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.

1.1.3 Viterbi, Complex Math, and CRC Unit II (VCU-II)

The VCU-II is the second-generation Viterbi, Complex Math, and CRC extension to the C28x CPU. The VCU-II extends the capabilities of the C28x CPU by adding registers and instructions to accelerate the performance of FFTs and communications-based algorithms. The C28x+VCU-II supports 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 1-2](#) shows a summary of the VCU performance for each of these operations.

Table 1-2. 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 ⁽¹⁾
Traceback per Stage	3 ⁽²⁾

⁽¹⁾ C28x CPU takes 15 cycles per butterfly.

⁽²⁾ C28x CPU takes 22 cycles per stage.

- **Cyclic Redundancy Check**

Cyclic redundancy check (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-bit, 16-bit, 24-bit, 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 in 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 1-3](#) shows a summary of the VCU operations enabled by the VCU.

Table 1-3. 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 and Accumulate (MAC)	2p	32 + 32 = 32-bit, 16 x 16 = 32-bit
RPT MAC	2p+N	Repeat MAC. Single cycle after the first operation.

System Control

This chapter explains system control and interrupts found on this MCU. The system control module configures and manages the overall operation of the device and provides information about the device status. Configurable features in system control include reset control, NMI operation, power control, clock control, and low-power modes.

Topic	Page
2.1 Introduction	83
2.2 System Control Functional Description	83
2.3 Resets	84
2.4 Peripheral Interrupts	86
2.5 Exceptions and Non-Maskable Interrupts	98
2.6 Safety Features	99
2.7 Clocking	103
2.8 32-Bit CPU Timers 0/1/2	113
2.9 Watchdog Timers	114
2.10 Low Power Modes	117
2.11 Memory Controller Module	119
2.12 Flash and OTP Memory	128
2.13 Dual Code Security Module (DCSM)	140
2.14 Registers	152

2.1 Introduction

On this device, the CPU1 subsystem acts as a master, and by default (upon reset), it owns all the configuration and control. Through software running on CPU1, peripherals and I/Os can be configured to be accessible by the CPU2 subsystem and the configuration so chosen could be locked.

The PLL clock configuration is also owned by the CPU1 subsystem by default, but a clock control semaphore is provided by which CPU2 can grab access to the clock configuration registers.

Each CPU has its own NMI module to handle different exceptions during run time. If the NMI was on CPU1, any NMI exception that is unhandled before the NMI Watchdog (NMIWD) timer expiration will reset the entire device. If the NMI was on the CPU2 subsystem, then the CPU2 subsystem alone will be reset, in which case the CPU1 subsystem will be informed by another NMI that the CPU2 subsystem was reset because of NMIWD timer expiration.

Each CPU subsystem has its own watchdog timer module for software to use. Watchdog timer expiration on CPU2 will reset the CPU2 subsystem alone when configured to generate a reset, but watchdog timer expiration on CPU1 will reset the entire device.

Except for a CPU2 standalone internal reset such as CPU2.NMIWD or CPU2.WD each time the device is reset, the CPU2 subsystem will be held under reset until the CPU1 subsystem brings it out of reset. This is done by the boot ROM software running on the CPU1 core.

The register space of the device system control module is divided into three categories and will be explained further in this chapter. They are:

1. System Control Device Configuration Registers (DEV_CFG_REGS). These registers are mapped to CPU1 only. The base address of these registers on the CPU1 address space begins at 0x5D000.
2. System Control Clock Configuration Registers (CLK_CFG_REGS). These registers are mapped to both CPU1 and CPU2 address space but access control is based on a Clock Control Semaphore register. The base address of these registers on both the CPU subsystems begins at 0x5D200.
3. System control CPU Subsystem Registers (CPU_SYS_REGS). These registers are mapped to both the CPU subsystems. The base address of these registers on both the CPU subsystems begins at 0x5D300.

This chapter explains the system control module on both the CPU subsystems.

2.2 System Control Functional Description

The system control module provides the following capabilities:

- Device identification and configuration registers
- Reset control
- Exceptions and Interrupt control
- Safety and error handling features of the device
- Power control
- Clock control
- Low Power modes
- Security module
- Inter-Processor Communication (IPC)

2.2.1 Device Identification

Device identification registers provide information on device class, device family, revision, part number, pin count, operating temperature range, package type, pin count, and device qualification status.

All of the device information is part of the DEV_CFG_REGS space and is accessible only by the software running on the CPU1 subsystem.

The control subsystem device identification registers are: PARTIDL, PARTIDH, and REVID.

2.2.2 Device Configuration Registers

Several registers provide users with configuration information for debug and identification purposes on this MCU. This information includes the features of the peripheral and how much RAM and FLASH memory is available on this part.

These registers are part of DEV_CFG_REGS space and are accessible only by the software running on the CPU1 subsystem.

- DC0 – DC20: Device Configuration or Capabilities registers.
If a particular bit in these registers is set to '1' then the associated/feature or module is available in the device.
- PERCNF: Peripheral configuration register.
This register configures ADC capabilities, and enables or disables the USB internal PHY.

2.3 Resets

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

2.3.1 Reset Sources

Table 2-1 summarizes the various reset signals and their effect on the device.

Table 2-1. Reset Signals

Reset Source	CPU1 Core Reset (C28x, TMU, FPU, VCU)	CPU1 Peripheral s Reset	CPU2 Core Reset (C28x, TMU, FPU, VCU)	CPU2 Peripheral s Reset	CPU2 Held In Reset	JTAG / Debug Logic Reset	IOs	XRS Output
POR	Yes	Yes	Yes	Yes	Yes	Yes	Hi-Z	Yes
XRS Pin	Yes	Yes	Yes	Yes	Yes	Yes	Hi-Z	-
CPU1.WDRS	Yes	Yes	Yes	Yes	Yes	Yes	Hi-Z	Yes
CPU1.NMIWDRS	Yes	Yes	Yes	Yes	Yes	Yes	Hi-Z	Yes
CPU1.SYSRS (Debugger Reset)	Yes	Yes	Yes	Yes	Yes	No	Hi-Z	No
CPU1.SCCRESET	Yes	Yes	Yes	Yes	Yes	Yes	Hi-Z	No
CPU2.SYSRS (Debugger Reset)	No	No	Yes	Yes	No	No	-	No
CPU2.WDRS	No	No	Yes	Yes	No	No	-	No
CPU2.NMIWDRS	No	No	Yes	Yes	No	No	-	No
CPU2.SCCRESET	No	No	Yes	Yes	No	Yes	-	No
HIBRESET	Yes	Yes	Yes	Yes	Yes	Yes	Isolated	No
CPU1.HWBISTRs	Yes	No	No	No	No	No	-	No
CPU2.HWBISTRs	No	No	Yes	No	No	No	-	No
TRST	No	No	No	No	No	Yes	-	No

The resets can be divided into a few groups:

- Chip-level resets (XRS, POR, CPU1.WDRS, and CPU1.NMIWDRS), which reset all or almost all of the device.
- System resets (CPU1.SYSRS and CPU1.SCCRESET), which reset a large subset of the device but maintain some system-level configuration.
- CPU2 subsystem resets (CPU2.SYSRS, CPU2.WDRS, CPU2.NMIWDRS, and CPU2.SCCRESET), which reset only CPU2 and its peripherals.
- Special resets (HIBRESET, CPU1.HWBISTRs, CPU2.HWBISTRs, and TRST), which enable specific device functions.

Whenever the CPU1 subsystem is reset, CPU2 and its peripherals are also reset, and CPU2 is held in reset. CPU1 brings it out of reset by writing to the CPU2RECTL register. This is normally done by the boot ROM. For more details on the boot process, refer to the *ROM Code and Peripheral Booting* chapter.

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 register. Each CPU has its own RESC register, referred to as CPU1.RESC and CPU2.RESC.

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

After a POR, XRS, CPU1.WDRS, CPU1.NMIWDRS, or HIBRESET, the boot ROMs will clear all of the system and message RAMs on both CPUs. After a CPU2.WDRS or CPU2.NMIWDRS, CPU2's boot ROM will clear all of the CPU2 system and message RAMs.

2.3.2 External Reset (\overline{XRS})

The external reset (\overline{XRS}) is the main chip-level reset for the device. It resets both CPUs, all peripherals and I/O pin configurations, and most of the system control registers. It also holds CPU2 in reset. 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. In hibernate mode, toggling \overline{XRS} will produce a HIBRESET.

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.

2.3.3 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, \overline{XRS} 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), the X1 clock counter register (X1CNT), and the hibernate configuration registers (HIBBOOTMODE, IORESTOREADDR, and LPMCR.M0M1MODE).

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

2.3.4 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, each CPU has its own subsystem reset, which can be triggered by a debugger using Code Composer Studio. CPU2's subsystem reset (CPU2. \overline{SYSRS}) resets only CPU2, its peripherals, and its clock gating and LPM configuration. It does not hold CPU2 in reset. CPU1's subsystem reset (CPU1. \overline{SYSRS}) resets CPU1, its peripherals, many system control registers (including its clock gating and LPM configuration and the peripherals' CPU ownership), and all I/O pin configurations. It also produces a CPU2. \overline{SYSRS} and holds CPU2 in reset.

Neither \overline{SYSRS} resets 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 2.3.3](#)).

2.3.5 Watchdog Reset (\overline{WDRS})

Each CPU has a watchdog timer that can optionally trigger a reset that lasts for 512 INTOSC1 cycles. CPU1's watchdog reset (CPU1. \overline{WDRS}) produces an \overline{XRS} . CPU2's watchdog reset (CPU2. \overline{WDRS}) produces a CPU2. \overline{SYSRS} and triggers an NMI on CPU1.

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

2.3.6 NMI Watchdog Reset ($\overline{\text{NMIWDRS}}$)

Each CPU has a non-maskable interrupt (NMI) module that detects hardware errors in the system. Each 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. CPU1's NMI watchdog reset (CPU1.NMIWDRS) produces an $\overline{\text{XRS}}$. CPU2's NMI watchdog reset (CPU2.NMIWDRS) produces a CPU2.SYSRS and triggers an NMI on CPU1.

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

2.3.7 DCSM Safe Code Copy Reset ($\overline{\text{SCCRESET}}$)

Each CPU 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. CPU1's security reset (CPU1.SCCRESET) is similar to a CPU1.SYSRS, and CPU2's security reset (CPU2.SCCRESET) is similar to a CPU2.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.

2.3.8 Hibernate Reset ($\overline{\text{HIBRESET}}$)

Hibernate is a chip-level low-power mode that gates power to large portions of the device. Waking up from hibernate involves a special reset ($\overline{\text{HIBRESET}}$). This reset is similar to a POR except that the I/O pins remain isolated and the $\overline{\text{XRS}}$ pin is not toggled. (An external $\overline{\text{XRS}}$ toggle during hibernate will trigger a $\overline{\text{HIBRESET}}$). I/O isolation is disabled in software as part of a special boot ROM flow. For more information on hibernate, refer to [Section 2.10](#).

After a hibernate reset, the HIBRESETn bit in RESC is set. This bit is then cleared by the boot ROM.

2.3.9 Hardware BIST Reset ($\overline{\text{HWBISTRs}}$)

Each CPU has a Hardware Built-In Self Test (HWBIST) module that tests the functionality of the CPU. At the end of the test, it resets the CPU to return it to a working state. This reset ($\overline{\text{HWBISTRs}}$) only affects the CPU itself. The peripherals and system control remain as previously configured. The CPU state is restored in software as part of a special boot ROM flow. For more information on the HWBIST flow, refer to the *HWBIST* chapter.

After a HWBIST reset, the HWBISTRn bit in RESC is set. This bit is then cleared by the boot ROM.

2.3.10 Test Reset ($\overline{\text{TRST}}$)

The ICEPick debug module and associated JTAG logic has its own reset ($\overline{\text{TRST}}$) which is controlled by a dedicated pin. This reset is normally active unless the user connects a debugger to the device. For more information on the debug module, see the TI Processors Wiki page on ICEPick:

<http://processors.wiki.ti.com/index.php/ICEPICK>.

The $\overline{\text{TRST}}$ does not have a normal RESC bit, but the TRSTn_pin_status bit indicates the state of the pin.

2.4 Peripheral Interrupts

This section explains the peripheral interrupt handling on the device. Non-maskable interrupts are covered in [Section 2.5](#). Software interrupts and emulation interrupts are not covered in this document. For information on those, see the *TMS320C28x CPU and Instruction Set Reference Guide* ([SPRU430](#)).

2.4.1 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.

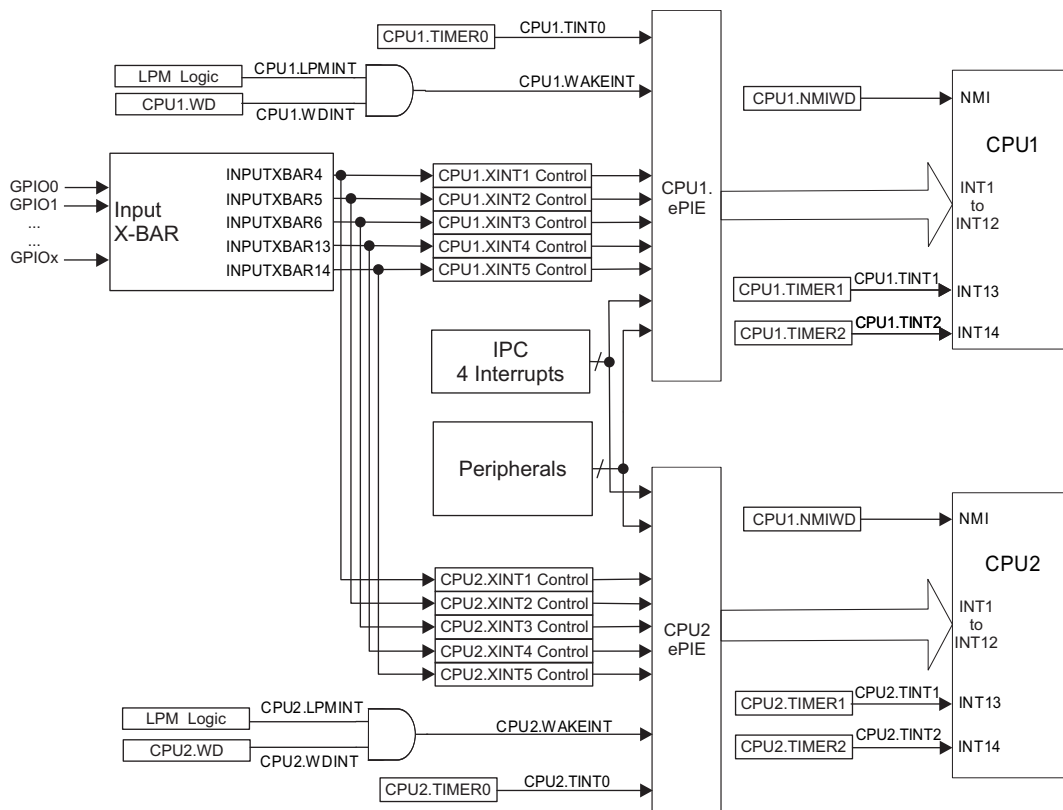
2.4.2 Interrupt Architecture

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 2-1 shows the interrupt architecture for this device.

Figure 2-1. Device Interrupt Architecture



2.4.2.1 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.

2.4.2.2 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 2.4.4](#) contains procedures for safely modifying the PIE configuration once interrupts have been enabled.

2.4.2.3 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 instruction. In C code, controlSUITE's DINT and EINT macros can be used for this purpose.

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

2.4.2.4 Dual-CPU Interrupt Handling

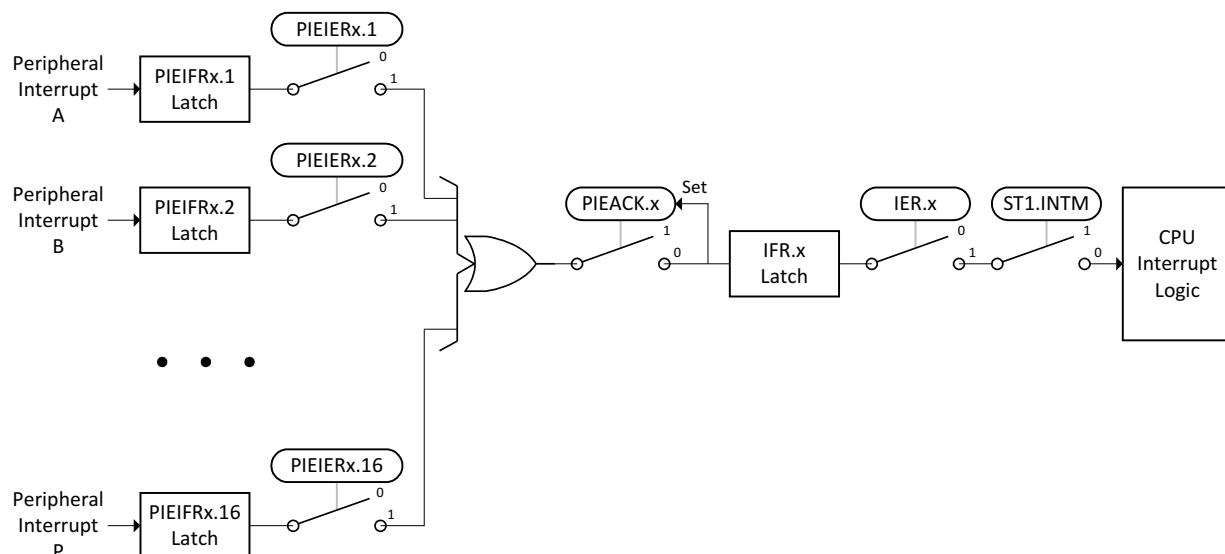
Each CPU has its own PIE. Both PIEs must be configured independently.

Some interrupts come from shared peripherals that can be owned by either CPU, such as the ADCs and SPIs. These interrupts are sent to both PIEs regardless of the peripheral's ownership. Thus, a peripheral owned by one CPU can cause an interrupt on the other CPU if that interrupt is enabled in the other CPU's PIE.

2.4.3 Interrupt Entry Sequence

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

Figure 2-2. 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.

2.4.4 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.

2.4.4.1 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 2-2](#).
4. Set the appropriate PIEIERx bit for each interrupt. The PIE group and channel assignments can be found in [Table 2-2](#).
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.

2.4.4.2 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* ([SPRU514](#)). 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* ([SPRU430](#)).

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.

2.4.4.3 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.

2.4.4.4 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 controlSUITE and on the TI Processors wiki:

http://processors.wiki.ti.com/index.php/Interrupt_Nesting_on_C28x

2.4.5 PIE Channel Mapping

Table 2-2 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 2-2. 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	ADCD1	TIMER0	WAKE	-	-	-	-	IPC0	IPC1	IPC2	IPC3
INT2.y	EPWM1_TZ	EPWM2_TZ	EPWM3_TZ	EPWM4_TZ	EPWM5_TZ	EPWM6_TZ	EPWM7_TZ	EPWM8_TZ	EPWM9_TZ	EPWM10_TZ	EPWM11_TZ	EPWM12_TZ	-	-	-	-
INT3.y	EPWM1	EPWM2	EPWM3	EPWM4	EPWM5	EPWM6	EPWM7	EPWM8	EPWM9	EPWM10	EPWM11	EPWM12	-	-	-	-
INT4.y	ECAP1	ECAP2	ECAP3	ECAP4	ECAP5	ECAP6	-	-	-	-	-	-	-	-	-	-
INT5.y	EQEP1	EQEP2	EQEP3	-	-	-	-	-	SD1	SD2	-	-	-	-	-	-
INT6.y	SPIA_RX	SPIA_TX	SPIB_RX	SPIB_TX	MCBSPA_RX	MCBSPA_TX	MCBSPB_RX	MCBSPB_TX	SPIC_RX	SPIC_TX	-	-	-	-	-	-
INT7.y	DMA_CH1	DMA_CH2	DMA_CH3	DMA_CH4	DMA_CH5	DMA_CH6	-	-	-	-	-	-	-	-	-	-
INT8.y	I2CA	I2CA_FIFO	I2CB	I2CB_FIFO	SCIC_RX	SCIC_TX	SCID_RX	SCID_TX	-	-	-	-	-	-	UPPA (CPU1 only)	-
INT9.y	SCIA_RX	SCIA_TX	SCIB_RX	SCIB_TX	CANA_1	CANA_2	CANB_1	CANB_2	-	-	-	-	-	-	USBA (CPU1 only)	-
INT10.y	ADCA_EVT	ADCA2	ADCA3	ADCA4	ADCB_EVT	ADCB2	ADCB3	ADCB4	ADCC_EVT	ADCC2	ADCC3	ADCC4	ADCD_EVT	ADCD2	ADCD3	ADCD4
INT11.y	CLA1_1	CLA1_2	CLA1_3	CLA1_4	CLA1_5	CLA1_6	CLA1_7	CLA1_8	-	-	-	-	-	-	-	-
INT12.y	XINT3	XINT4	XINT5	-	-	VCU	FPU_OVERFLOW	FPU_UNDERFLOW	EMIF_ERROR	RAM_CORRECTABLE_ERROR	FLASH_CORRECTABLE_ERROR	RAM_ACCESS_VIOLATION	SYS_PLL_SLIP	AUX_PLL_SLIP	CLA_OVERFLOW	CLA_UNDERFLOW

Note: Cells marked "-" are Reserved

2.4.6 Vector Tables

Table 2-3 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.

Table 2-3. 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 2-4 shows the Pie vector table.

Table 2-4. PIE Interrupt Vectors

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
PIE Group 1 Vectors - Muxed into CPU INT1						
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	ADCD1 interrupt	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	IPC1 interrupt	5	13
INT1.14	133	0x0000 0E0A	2	IPC2 interrupt	5	14
INT1.15	134	0x0000 0E0C	2	IPC3 interrupt	5	15
INT1.16	135	0x0000 0E0E	2	IPC4 interrupt	5	16 (Lowest)
PIE Group 2 Vectors - Muxed into CPU INT2						
INT2.1	40	0x0000 0D50	2	EPWM1_TZ interrupt	6	1 (Highest)
INT2.2	41	0x0000 0D52	2	EPWM2_TZ interrupt	6	2
INT2.3	42	0x0000 0D54	2	EPWM3_TZ interrupt	6	3
INT2.4	43	0x0000 0D56	2	EPWM4_TZ interrupt	6	4
INT2.5	44	0x0000 0D58	2	EPWM5_TZ interrupt	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6_TZ interrupt	6	6
INT2.7	46	0x0000 0D5C	2	EPWM7_TZ interrupt	6	7
INT2.8	47	0x0000 0D5E	2	EPWM8_TZ interrupt	6	8
INT2.9	136	0x0000 0E10	2	EPWM9_TZ interrupt	6	9
INT2.10	137	0x0000 0E12	2	EPWM10_TZ interrupt	6	10
INT2.11	138	0x0000 0E14	2	EPWM11_TZ interrupt	6	11
INT2.12	139	0x0000 0E16	2	EPWM12_TZ interrupt	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)
PIE Group 3 Vectors - Muxed into CPU INT3						
INT3.1	48	0x0000 0D60	2	EPWM1 interrupt	7	1 (Highest)
INT3.2	49	0x0000 0D62	2	EPWM2 interrupt	7	2

Table 2-4. PIE Interrupt Vectors (continued)

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
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
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	EPWM9 interrupt	7	9
INT3.10	145	0x0000 0E22	2	EPWM10 interrupt	7	10
INT3.11	146	0x0000 0E24	2	EPWM11 interrupt	7	11
INT3.12	147	0x0000 0E26	2	EPWM12 interrupt	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)
PIE Group 4 Vectors - Muxed into CPU INT4						
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	Reserved	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	Reserved	8	14
INT4.15	158	0x0000 0E3C	2	Reserved	8	15
INT4.16	159	0x0000 0E3E	2	Reserved	8	16 (Lowest)
PIE Group 5 Vectors - Muxed into CPU INT5						
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	EQEP3 interrupt	9	3
INT5.4	67	0x0000 0D86	2	Reserved	9	4
INT5.5	68	0x0000 0D88	2	Reserved	9	5
INT5.6	69	0x0000 0D8A	2	Reserved	9	6
INT5.7	70	0x0000 0D8C	2	Reserved	9	7
INT5.8	71	0x0000 0D8E	2	Reserved	9	8
INT5.9	160	0x0000 0E40	2	SD1 interrupt	9	9
INT5.10	161	0x0000 0E42	2	SD2 interrupt	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	Reserved	9	13

Table 2-4. PIE Interrupt Vectors (continued)

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT5.14	165	0x0000 0E4A	2	Reserved	9	14
INT5.15	166	0x0000 0E4C	2	Reserved	9	15
INT5.16	167	0x0000 0E4E	2	Reserved	9	16 (Lowest)
PIE Group 6 Vectors - Muxed into CPU INT6						
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	MCBSPA_RX interrupt	10	5
INT6.6	77	0x0000 0D9A	2	MCBSPA_TX interrupt	10	6
INT6.7	78	0x0000 0D9C	2	MCBSPB_RX interrupt	10	7
INT6.8	79	0x0000 0D9E	2	MCBSPB_TX interrupt	10	8
INT6.9	168	0x0000 0E50	2	SPIC_RX interrupt	10	9
INT6.10	169	0x0000 0E52	2	SPIC_TX interrupt	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)
PIE Group 7 Vectors - Muxed into CPU INT7						
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	Reserved	11	11
INT7.12	179	0x0000 0E66	2	Reserved	11	12
INT7.13	180	0x0000 0E68	2	Reserved	11	13
INT7.14	181	0x0000 0E6A	2	Reserved	11	14
INT7.15	182	0x0000 0E6C	2	Reserved	11	15

Table 2-4. PIE Interrupt Vectors (continued)

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT7.16	183	0x0000 0E6E	2	Reserved	11	16 (Lowest)
PIE Group 8 Vectors - Muxed into CPU INT8						
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	I2CB interrupt	12	3
INT8.4	91	0x0000 0DB6	2	I2CB_FIFO interrupt	12	4
INT8.5	92	0x0000 0DB8	2	SCIC_RX interrupt	12	5
INT8.6	93	0x0000 0DBA	2	SCIC_TX interrupt	12	6
INT8.7	94	0x0000 0DBC	2	SCID_RX interrupt	12	7
INT8.8	95	0x0000 0DBE	2	SCID_TX interrupt	12	8
INT8.9	184	0x0000 0E70	2	Reserved	12	9
INT8.10	185	0x0000 0E72	2	Reserved	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	Reserved	12	13
INT8.14	189	0x0000 0E7A	2	Reserved	12	14
INT8.15	190	0x0000 0E7C	2	UPPA interrupt (CPU1 only)	12	15
INT8.16	191	0x0000 0E7E	2	Reserved	12	16 (Lowest)
PIE Group 9 Vectors - Muxed into CPU INT9						
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	DCANA_1 interrupt	13	5
INT9.6	101	0x0000 0DCA	2	DCANA_2 interrupt	13	6
INT9.7	102	0x0000 0DCC	2	DCANB_1 interrupt	13	7
INT9.8	103	0x0000 0DCE	2	DCANB_2 interrupt	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	USBA interrupt (CPU1 only)	13	15
INT9.16	199	0x0000 0E8E	2	Reserved	13	16 (Lowest)
PIE Group 10 Vectors - Muxed into CPU INT10						

Table 2-4. PIE Interrupt Vectors (continued)

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT10.1	104	0x0000 0DD0	2	ADCA_EVT 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_EVT interrupt	14	5
INT10.6	109	0x0000 0DDA	2	ADCB2 interrupt	14	6
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_EVT 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	ADCD_EVT interrupt	14	13
INT10.14	205	0x0000 0E9A	2	ADCD2 interrupt	14	14
INT10.15	206	0x0000 0E9C	2	ADCD3 interrupt	14	15
INT10.16	207	0x0000 0E9E	2	ADCD4 interrupt	14	16 (Lowest)
PIE Group 11 Vectors - Muxed into CPU INT11						
INT11.1	112	0x0000 0DE0	2	CLA1_1 interrupt	15	1 (Highest)
INT11.2	113	0x0000 0DE2	2	CLA1_2 interrupt	15	2
INT11.3	114	0x0000 0DE4	2	CLA1_3 interrupt	15	3
INT11.4	115	0x0000 0DE6	2	CLA1_4 interrupt	15	4
INT11.5	116	0x0000 0DE8	2	CLA1_5 interrupt	15	5
INT11.6	117	0x0000 0DEA	2	CLA1_6 interrupt	15	6
INT11.7	118	0x0000 0DEC	2	CLA1_7 interrupt	15	7
INT11.8	119	0x0000 0DEE	2	CLA1_8 interrupt	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)
PIE Group 12 Vectors - Muxed into CPU INT12						
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	VCU interrupt	16	6
INT12.7	126	0x0000 0DFC	2	FPU_OVERFLOW interrupt	16	7
INT12.8	127	0x0000 0DFE	2	FPU_UNDERFLOW interrupt	16	8

Table 2-4. PIE Interrupt Vectors (continued)

Name	Vector ID	Address	Size (x16)	Description	Core priority	ePIE group Priority
INT12.9	216	0x0000 0EB0	2	EMIF_ERROR interrupt	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	SYS_PLL_SLIP interrupt	16	13
INT12.14	221	0x0000 0EBA	2	AUX_PLL_SLIP interrupt	16	14
INT12.15	222	0x0000 0EBC	2	CLA_OVERFLOW interrupt	16	15
INT12.16	223	0x0000 0EBE	2	CLA_UNDERFLOW interrupt	16	16 (Lowest)

2.5 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.

2.5.1 Configuring and Using NMIs

Each CPU has its own NMI module. 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.

2.5.2 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.

2.5.3 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.

2.5.3.1 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 NMIs are fired to both CPUs. For more information on missing clock detection, see [Section 2.6.2](#).

2.5.3.2 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 2.11.1.8](#).

2.5.3.3 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 [Section 2.12.10](#).

2.5.3.4 NMI Vector Fetch Mismatch

Each CPU's Peripheral Interrupt Expansion module (PIE) has redundant vector tables. If a mismatch in these tables is detected during a vector fetch, a user-specified error handler is run instead of the ISR. If the vector fetch was caused by an NMI, a second NMI is fired to the other CPU. Mismatches for other interrupts do not trigger an NMI. For more information about the vector address check, see [Section 2.6.4](#).

2.5.3.5 CPU2 Watchdog or NMI Watchdog Reset

A watchdog reset or NMI watchdog reset on CPU2 will trigger an NMI on CPU1. Since a CPU1 reset also resets CPU2, this NMI source is not available on CPU2.

2.5.4 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* ([SPRU430](#)).

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.

2.6 Safety Features

This section gives details on different modules or features that safeguard device operation during run-time, and catch serious errors that can occur. Also included are details on hardware behavior when any of the serious errors occurs in the device.

2.6.1 Write Protection on Registers

2.6.1.1 LOCK Protection on System Configuration Registers

Several system configuration registers are protected from spurious CPU writes by "LOCK" registers. Once these associated LOCK register bits are set the respective locked registers can no longer be modified by software. See specific register descriptions for details.

2.6.1.2 EALLOW Protection

Several control registers are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates the state of protection as shown in Table 2-5.

Table 2-5. Access to EALLOW-Protected Registers

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed ⁽¹⁾	Allowed
1	Allowed	Allowed	Allowed	Allowed

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

At reset, the EALLOW bit is cleared, enabling EALLOW protection. While protected, all writes to protected registers by the CPU are ignored and only CPU reads, JTAG reads, and JTAG writes are allowed. If this bit is set, 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.

2.6.2 Missing Clock Detection Logic

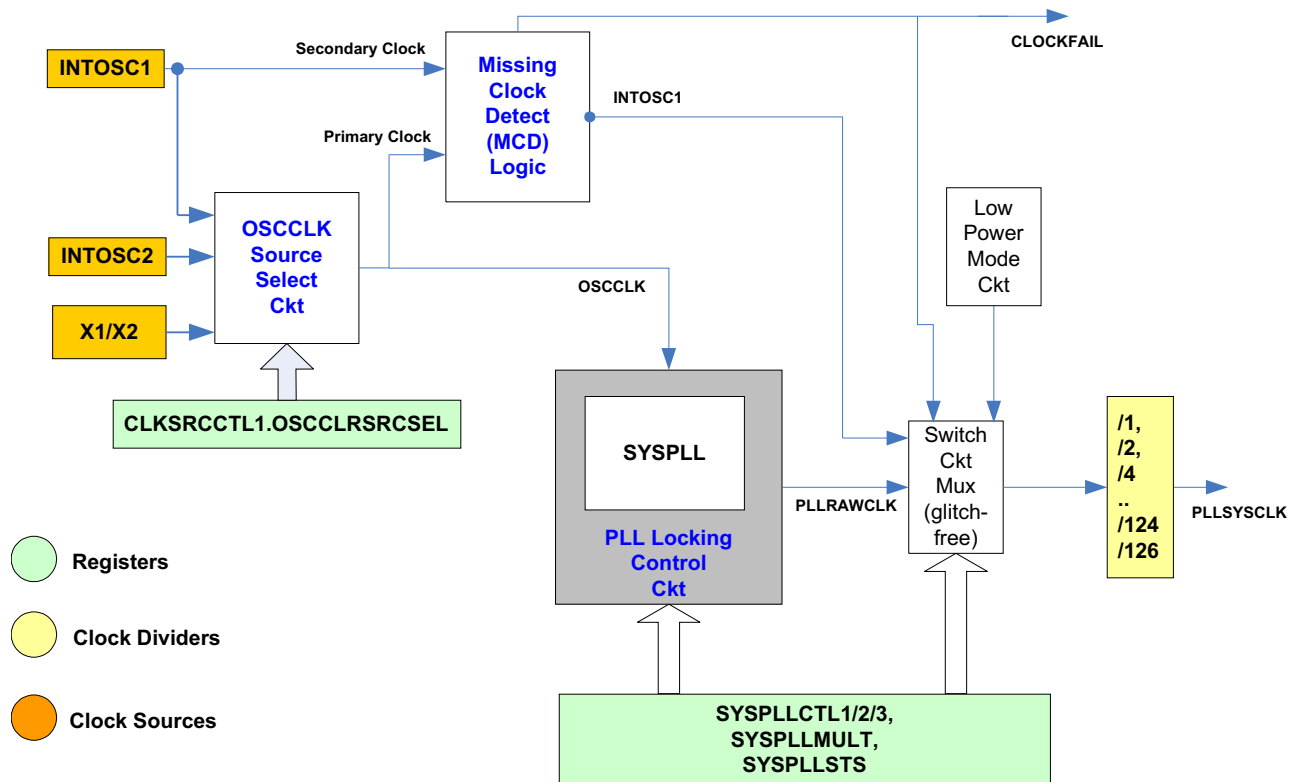
The missing clock detect (MCD) logic detects OSCCLK failure, using INTOSC1 as the reference clock source. This circuit only detects complete loss of OSCCLK and doesn't do any detection of frequency drift on the OSCCLK.

This circuit monitors the OSCCLK (primary clock) using the 10 MHz clock provided by the INTOSC1 (secondary clock) as a backup clock. This circuit functions as below:

1. The primary clock (OSCCLK) clock keeps ticking a 7-bit counter (named as MCDPCNT). This counter is asynchronously reset with \overline{XRS} .
2. The secondary clock (INTOSC1) clock keeps ticking a 13-bit counter (named as MCDSCNT). This counter is asynchronously reset with \overline{XRS} .
3. Each time MCDPCNT overflows, the MCDSCNT counter is reset. Thus, if OSCCLK is present or not slower than INTOSC1 by a factor of 64, MCDSCNT will never overflow.
4. If OSCCLK stops for some reason, or is slower than INTOSC1 by at least a factor of 64, the MCDSCNT will overflow and a missing clock condition will be detected on OSCCLK.
5. The above check is continuously active, unless the MCD is disabled using MCDPCR register (by making the MCLKOFF bit 1)
6. If the circuit ever detects a missing OSCCLK, the following occurs:
 - The MCDSTS flag is set
 - The MCDSCNT counter is frozen to prevent further missing clock detection
 - The CLOCKFAIL signal goes high, which generates TRIP events to PWM modules and fires NMIs to CPU1.NMIWD and CPU2.NMIWD.
 - PLL is forcefully bypassed and OSCCLK is switched to INTOSC1 (after the PLLSYSCLK divider). PLLMULT is zeroed out automatically in this case.
 - While the MCDSTS bit is set, the OSCCLKSRCSEL bits have no effect and OSCCLK is forcefully connected to INTOSC1.
 - PLLRAWCLK going to the system is switched to INTOSC1 automatically
7. If the MCLKCLR bit is written (this is a W=1 bit), MCDSTS bit will be cleared and OSCCLK source will be decided by the OSCCLKSRCSEL bits. Writing to MCLKCLR will also clear the MCDPCNT and MCDSCNT counters to allow the circuit re-evaluate missing clock detection. If user wants to lock the PLL after missing clock detection, he needs to first switch the clock source to INTOSC1 (using OSCCLKSRCSEL register), do a MCLKCLR and re-lock the PLL.
8. The MCD is enabled at power up. There is no support for missing clock detection if INTOSC2 is failed from the device power-up.

Figure 2-3 shows the missing clock logic functional flow.

Figure 2-3. Missing Clock Detection Logic



2.6.3 PLLSLIP Detection

The PLL SLIP detection on this device can detect if the PLL reference clock goes too high or too slow while PLL is locked. An interrupt to both the CPUs is triggered as shown in the ePIE table in [Section 2.4](#). Apart from the interrupt to both the CPUs, the PLLSTS.SLIP bit is set for user software to check the error.

The SLIP detection is available on both SYSPLL and AUXPLL.

2.6.4 CPU1 and CPU2 PIE Vector Address Validity Check

The ePIE vector table on each CPU is duplicated into these two parts:

- Main ePIE Vector Table mapped from 0xD00 to 0xEFF in the C28x memory space
- Redundant ePIE Vector Table mapped from 0x1000D00 to 0x1000EFF in the C28x memory space

Following is the behavior of accesses to the ePIE memories:

- Data Writes to Main Vector Table: Writes to both memories
- Data Writes to Redundant Vector Table: Writes only to the Redundant Vector Table
- Vector Fetch: Data from both the vector tables are compared
- Data Read: Can read the Main and Redundant vector table separately

On every vector fetch from the ePIE, a hardware comparison (no cycle penalty is incurred to do the comparison) of both the vector table outputs is performed and if there is a mismatch between the two vector table outputs, the following occurs:

1. If the PIEVERRADDR register (default value 0x3F FFFF) is not initialized, the default error handler at address 0x3FFFBE gets executed.

But, when the PIEVERRADDR register is initialized to the address of the user-defined routine, the user-defined routine is executed instead of the default error handler.

Note: Each CPU has its own copy of the PIE Vector Fetch Error Handler register (CPU1.PIEVERRADDR and CPU2.PIEVERRADDR).

2. Hardware also generates EPWM Trip signals which will trip the PWM outputs using TRIPIN15.
3. An NMI to the other CPU is sent if the current mismatch is during a vector fetch. For example, on an NMI vector fetch error for CPU2, an NMI is also fired to CPU1.NMIWD.

If there is no mismatch, the correct vector is jammed onto the C28 program control.

2.6.5 NMIWDs

Each CPU has user-programmable NMIWD period registers, in which users can set a limit on how much time they want to allocate for the device to acknowledge the NMI. If the NMI is not acknowledged, it will cause a device reset.

2.6.6 ECC and Parity Enabled RAMs, Shared RAMs Protection

RAM memories in each CPU subsystem are ECC and parity enabled. All single-bit errors in ECC RAM are auto corrected and an error counter is incremented every time a single bit error is incremented. If the error counter reaches a predefined user configured limit, then an interrupt is generated to each CPU. Refer to [Section 2.11](#) for more details on RAM errors.

All uncorrectable double-bit errors end up triggering an NMI to corresponding CPUs.

2.6.7 ECC Enabled Flash Memory

When ECC is programmed and enabled, flash single-bit errors are corrected automatically by ECC logic before giving data to the CPU, but they are not corrected in flash memory. Flash memory will still contain wrong data until another erase/program operation happens to correct the flash contents. Irrespective of whether the error interrupt is enabled or disabled, single-bit errors are always corrected before giving data to the CPU. When the interrupt is disabled, users can check the single-bit error counter register for any single-bit error occurrences. The error counter stops incrementing once its value is equal to the threshold+1. It is always suggested to set the threshold register to a non-zero value so that the error counter can increment. It is up to the user to decide the threshold value at which they have to reprogram the flash with the correct data.

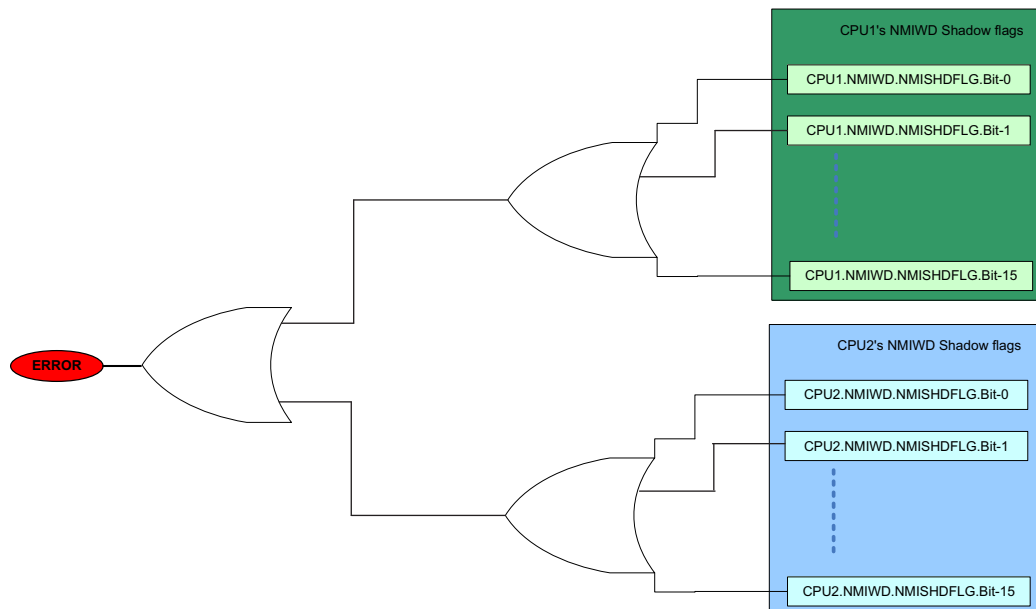
When ECC is programmed and enabled, flash uncorrectable errors end up triggering an NMI to the respective CPU. Please refer to [Section 2.11](#) for more details on flash error correction and error catching mechanisms.

2.6.8 Error Pin

The ERROR pin is an 'always output' pin and remains low until an error is detected inside the chip. On an error, the ERROR pin goes high until the corresponding internal error status flag for that error source is cleared. [Figure 2-4](#) shows the functionality of the ERROR pin.

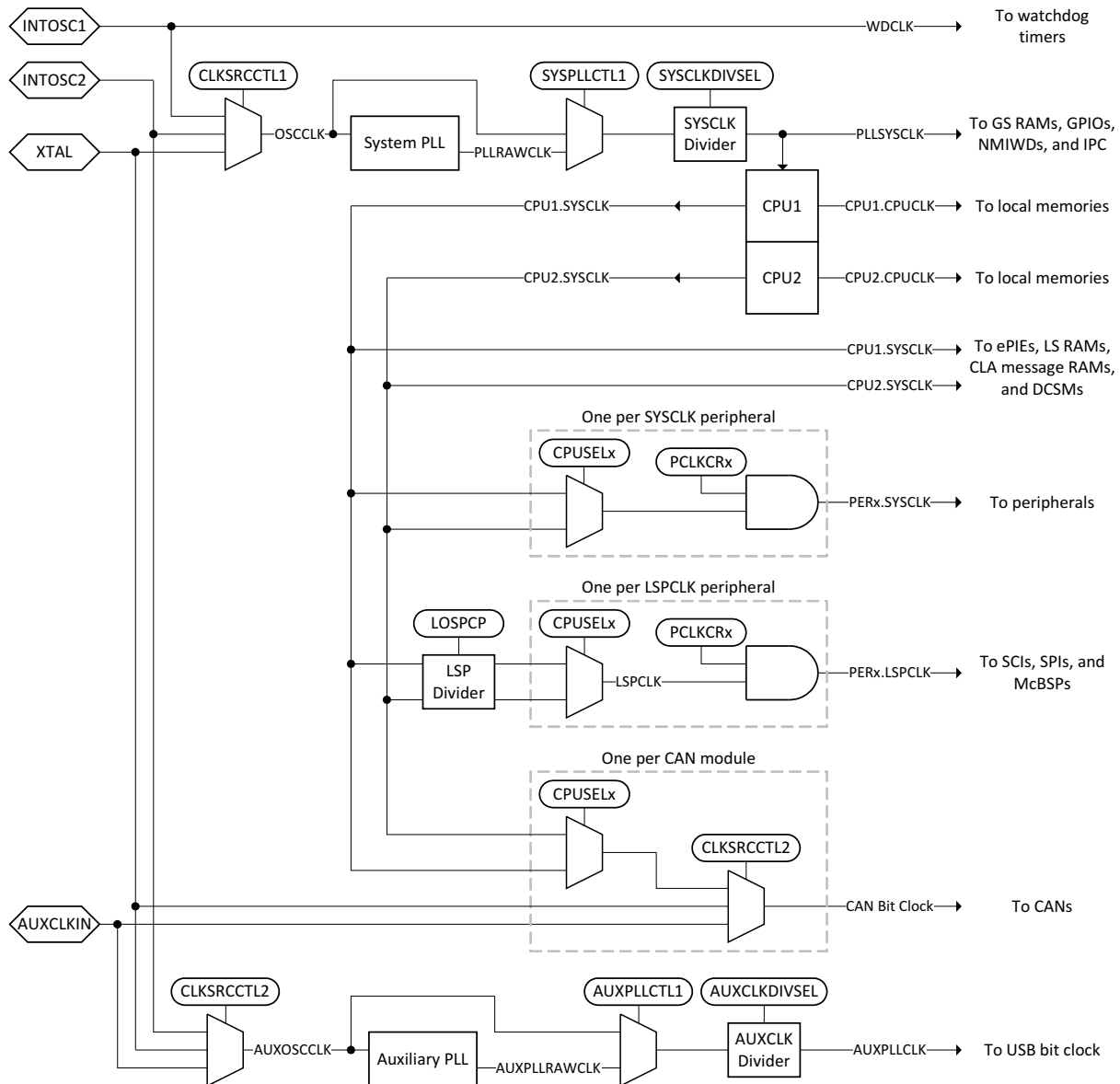
The error pin will be tri-stated until the chip power rails ramp up to the lower operational limit. As the ERROR pin is an active-high pin, users who care about the state of this pin during power-up should connect an external pull-down on this pin.

Figure 2-4. Error Pin Diagram



2.7 Clocking

This section explains the clock sources and clock domains on this device, and how to configure them for application use. [Figure 2-5](#) provides an overview of the device's clocking system.

Figure 2-5. Clocking System


Note: The default/2 divider for ePWMs and EMIFs is not shown.

2.7.1 Clock Sources

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

2.7.1.1 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 and USB, so an external clock must be used to support those features.

2.7.1.2 Backup Internal Oscillator (INTOSC1)

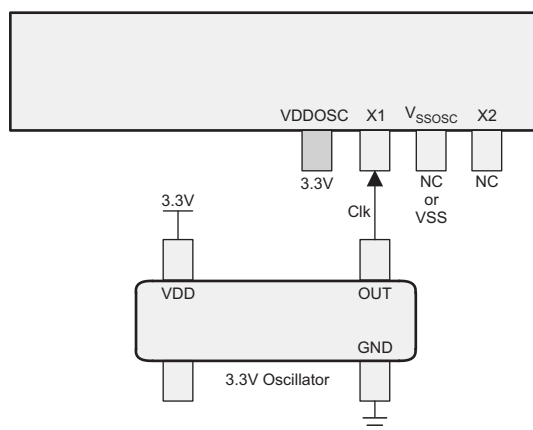
The device also includes a redundant on-chip 10 MHz oscillator (INTOSC1). INTOSC1 is a backup clock source that normally only clocks the watchdog timers and missing clock detection circuit (MCD). If MCD is enabled and a missing system clock is detected, the system PLL is bypassed and all system clocks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

2.7.1.3 External Oscillator (XTAL)

The dedicated X1 and X2 pins support an external clock source (XTAL), which can be used as the main system and auxiliary clock source. Frequency limits and timing requirements can be found in the device datasheet. Three types of external clock sources are supported:

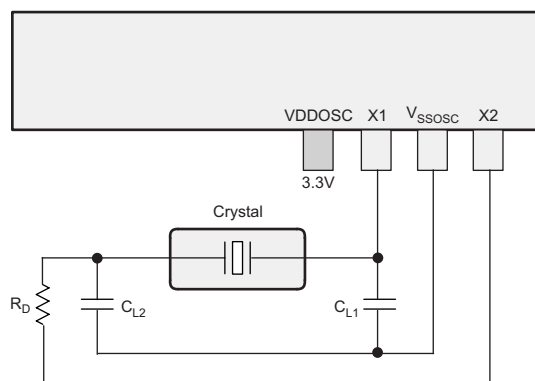
- A single-ended 3.3V external clock. The clock signal should be connected to X1 while X2 is left unconnected, as shown in [Figure 2-6](#).

Figure 2-6. Single-ended 3.3V External Clock

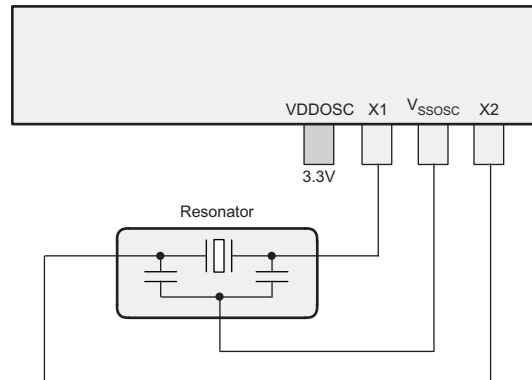


- An external crystal. The crystal should be connected across X1 and X2 with its load capacitors connected to VSSOSC as shown in [Figure 2-7](#).

Figure 2-7. External Crystal

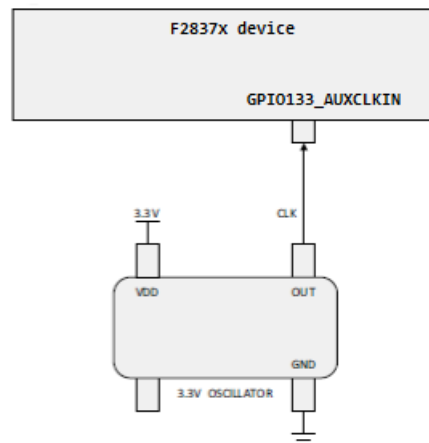


- An external resonator. The resonator should be connected across X1 and X2 with its ground connected to VSSOSC as shown in [Figure 2-8](#).

Figure 2-8. External Resonator


2.7.1.4 Auxiliary Clock Input (AUXCLKIN)

An additional external clock source is supported on GPIO133 (AUXCLKIN). This must be a single-ended 3.3V external clock. It can be used as the source for the USB and CAN bit clocks. Frequency limits and timing requirements can be found in the device datasheet. The external clock should be connected directly to the GPIO133 pin, as shown in [Figure 2-9](#).

Figure 2-9. AUXCLKIN


2.7.2 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.

2.7.2.1 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.

2.7.2.2 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 and slip detection. For configuration instructions, see [Section 2.7.6](#).

2.7.2.3 Auxiliary Oscillator Clock (AUXOSCCLK)

One of INTOSC2, XTAL, or AUXCLKIN may be chosen to be the auxiliary reference clock (AUXOSCCLK) for the USB module. (This selection does not affect the CAN bit clock, which uses AUXCLKIN directly). AUXOSCCLK may be used directly or fed through the auxiliary PLL to reach a higher frequency. At reset, AUXOSCCLK is connected to INTOSC2, but only an external oscillator can meet the USB timing requirements.

2.7.2.4 Auxiliary PLL Output Clock (AUXPLLRAWCLK)

The auxiliary PLL is used to generate a 60 MHz clock for the USB module. This PLL uses AUXOSCCLK as a reference, and features a fractional multiplier and slip detection. For configuration instructions, see [Section 2.7.6](#).

2.7.3 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.

2.7.3.1 System Clock (PLLSYSCLK)

The system control registers, GS RAMs, IPC module, GPIO qualification, and NMI watchdog timers have their 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 SYCLKDIVSEL register. PLLSYSCLK is gated in HALT mode.

2.7.3.2 CPU Clock (CPUCLK)

Each CPU has its own clock (CPU1.CPUCLK and CPU2.CPUCLK) which is used to clock the CPU, its coprocessors, its private RAMs (M0, M1, D0, and D1), and its boot ROM and flash wrapper. This clock is identical to PLLSYSCLK, but is gated when the CPU enters IDLE, STANDBY, or HALT mode.

2.7.3.3 CPU Subsystem Clock (SYSCLK and PERx.SYSCLK)

Each CPU provides a clock (CPU1.SYSCLK and CPU2.SYSCLK) to its CLA, DMA, and most owned peripherals. This clock is identical to PLLSYSCLK, but is gated when the CPU enters STANDBY or HALT mode.

Each peripheral clock can be connected to either CPU1.SYSCLK or CPU2.SYSCLK. This selection is made by CPU1 via the CPUSELx registers. Each peripheral clock also has its own independent clock gating which is controlled by the CPU's PCLKCRx registers. By default, the ePWM, EMIF1, and EMIF2 clocks each have an additional /2 divider, which is required to support CPU frequencies over 100 MHz. At slower CPU frequencies, these dividers can be disabled via the PERCLKDIVSEL register.

2.7.3.4 Low-Speed Peripheral Clock (LSPCLK and PERx.LSPCLK)

The SCI, SPI, and McBSP 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, SPI, and McBSP module's clock (PERx.LSPCLK) can be gated independently via the PCLKCRx registers.

2.7.3.5 USB Auxiliary Clock (AUXPLLCLK)

The USB module requires a fixed 60 MHz clock for bit sampling. Since the main system clock is usually not a multiple of 60 MHz, the correct frequency cannot be achieved with a simple divider. Instead, the USB clock is provided through an auxiliary clock path (AUXPLLCLK), which can use an independent clock source and PLL to generate the correct frequency.

USB clock tolerances are very tight. As stated in section 7.1.11 of the USB 2.0 specification, low-speed devices (1.50 Mb/s) have a tolerance of +/- 1.5% , while high-speed devices (12.000 Mb/s) have a tolerance of +/- 0.25%. Typically these tolerances are achieved by using an external crystal or resonator as the source for AUXOSCLK.

2.7.3.6 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 or AUXCLKIN via the CLKSRCCTL2 register. There is an independent selection for each CAN module.

2.7.3.7 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, XTAL, or AUXPLLCLK 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. Each CPU has its own independent CPU timers and TMR2CLKCTL register.

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

2.7.4 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, GPIO73. The available clock sources are PLLSYSCLK, PLLRAWCLK, CPU1.SYSCLK, CPU2.SYSCLK, AUXPLLRAWCLK, INTOSC1, and INTOSC2.

To use XCLKOUT, first select the clock source via the CLKSRCCTL3 register. Next, select the desired output divider via the XCLKOUTDIVSEL register. Finally, connect GPIO73 to mux channel 3 using the GPIO configuration registers.

2.7.5 Clock Connectivity

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

Table 2-6. Clock Connections Sorted by Clock Domain

Clock Domain	CPU1 Subsystem	CPU2 Subsystem	Shared Modules
CPUx.CPUCLK	CPU1 CPU1.VCU CPU1.FPU CPU1.TMU CPU1.M0 - M1 RAMs CPU1.D0 - D1 RAMs CPU1.BootROM CPU1.Flash	CPU2 CPU2.VCU CPU2.FPU CPU2.TMU CPU2.M0 - M1 RAMs CPU2.D0 - D1 RAMs CPU2.BootROM CPU2.Flash	
CPUx.SYSCLK	CPU1.ePIE CPU1.LS0 - LS5 RAMs CPU1.CLA1 Message RAMs CPU1.DCSM	CPU2.ePIE CPU2.LS0 - LS5 RAMs CPU2.CLA1 Message RAMs CPU2.DCSM	

Table 2-6. Clock Connections Sorted by Clock Domain (continued)

Clock Domain	CPU1 Subsystem	CPU2 Subsystem	Shared Modules
PLLSYSCLK	CPU1.NMIWD EMIF1	CPU2.NMIWD	GS0 - GS15 RAMs GPIO Input Sync and Qual IPC
PERx.SYSCLK	CPU1.CLA1 CPU1.DMA CPU1.Timer0 - 2 EMIF2 uPP A	CPU2.CLA1 CPU2.DMA CPU2.Timer0 - 2	ADCA - D CMPSS1 - 8 DACA - C ePWM1 - 12 eCAP1 - 6 eQEP1 - 3 I2CA - B McBSPA - B SDFM1 - 8
PERx.LSPCLK			McBSPA - B SCIA - D SPIA - C
CAN Bit Clock			CANA - B
AUXPLLCLK	USB		
WDCLK (INTOSC1)	CPU1.Watchdog	CPU2.Watchdog	

Table 2-7. Clock Connections Sorted by Module Name

Module Name	Clock Domain
ADCA - D	PERx.SYSCLK
Boot ROM	CPUx.CPUCLK
CANA - B	CAN Bit Clock
CLA	PERx.SYSCLK
CLA Message RAMs	CPUx.SYSCLK
CMPSS1 - 8	PERx.SYSCLK
CPU	CPUx.CPUCLK
CPU Timers	PERx.SYSCLK
D0 - D1 RAMs	CPUx.CPUCLK
DACA - C	PERx.SYSCLK
DCSM	CPUx.SYSCLK
DMA	PERx.SYSCLK
eCAP1 - 6	PERx.SYSCLK
EMIF1	PLLSYSCLK
EMIF2	PERx.SYSCLK
ePIE	CPUx.SYSCLK
ePWM	PERx.SYSCLK
eQEP1 - 3	PERx.SYSCLK
Flash	CPUx.CPUCLK
FPU	CPUx.CPUCLK
GS0 - GS15 RAMs	PLLSYSCLK
I2CA - B	PERx.SYSCLK
IPC	PLLSYSCLK
LS0 - LS5 RAMs	CPUx.SYSCLK

Table 2-7. Clock Connections Sorted by Module Name (continued)

Module Name	Clock Domain
M0 - M1 RAMs	CPUx.CPUCLK
McBSPA - B	PERx.LSPCLK
NMIWD	PLLSYSCLK
SCIA - D	PERx.LSPCLK
SDFM1 - 8	PERx.SYSCLK
SPIA - C	PERx.LSPCLK
TMU	CPUx.CPUCLK
uPP	PERx.SYSCLK
USB	AUXPLLCLK
VCU	CPUx.CPUCLK
Watchdog Timer	WDCLK (INTOSC1)

2.7.6 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. Is USB required?
4. What types of external oscillators or clock sources are available?

If CAN or USB 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.

2.7.6.1 Choosing PLL Settings

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

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

$$f_{\text{AUXPLLCLK}} = f_{\text{AUXOSCCLK}} * (\text{AUXPLLMULT.IMULT} + \text{AUXPLLMULT.FMULT}) / \text{AUXCLKDIVSEL.AUXPLLDIV}$$

where f_{OSCCLK} is the system oscillator clock frequency, $f_{\text{AUXOSCCLK}}$ is the auxiliary oscillator clock frequency, IMULT and FMULT are the integral and fractional parts of the multipliers, PLLSYSCLKDIV is the system clock divider, and AUXPLLDIV is the auxiliary 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 of 110 - 550 MHz.

NOTE: The system clock frequency (PLLSYSCLK) may not exceed the limit specified in the datasheet. This limit does not allow for oscillator tolerance.

The clock source and PLL configuration registers are shared between the two CPUs. Register access is controlled via a semaphore, which is described in the *Inter-Processor Communication* chapter.

2.7.6.2 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.
2. Set up the system PLL if desired:
 - (a) To reduce inrush current: Set the system clock divider to the desired value plus one by writing to SYSCLKDIVSEL.PLLSYSCLKDIV.
 - (b) Set the integral and fractional multipliers by simultaneously writing them both to SYSPLLMULT. This will automatically enable the PLL. Be sure that the product of OSCCLK and the multiplier is between 110 and 550 MHz.
 - (c) Wait for the PLL to lock by polling the SYSPLLSTS.LOCKS bit. This will take 16 μ s plus 1024 OSCCLK cycles.
 - (d) Repeat steps b and c to guarantee correct PLL lock.
 - (e) Connect the PLL output clock (PLLRAWCLK) to the system by writing a 1 to SYSPLLCTL1.PLLCLKEN.
 - (f) Set the desired system clock divider by writing to SYSCLKDIVSEL.PLLSYSCLKDIV.
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. If the CPU frequency is 100 MHz or less, set the ePWM and EMIF clock dividers to /1 by writing to PERCLKDIVSEL.
6. 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.

NOTE: If the CPU2 changes the OSCCLK source, it will not automatically bypass the PLL. The CPU2 must manually bypass the PLL first by writing a 0 to SYSPLLCTL1.PLLCLKEN.

2.7.6.3 USB Auxiliary Clock Setup

If USB functionality is needed, the auxiliary clock (AUXPLLCLK) must be configured to produce 60 MHz. The procedure is similar to the system clock setup:

1. Select the reference clock source (AUXOSCCLK) by writing to CLKSRCCTL2.AUXOSCCLKSRCSEL.
2. Wait two AUXOSCCLK cycles.
3. Set up the auxiliary PLL. If the PLL is not needed, bypass it and power it down by writing a 0 to AUXPLLCTL1.PLEN. To use the PLL:
 - (a) Set the desired auxiliary clock divider by writing to AUXCLKDIVSEL.AUXPLLDIV.
 - (b) Set the integral and fractional multipliers simultaneously. This will automatically enable the PLL. Be sure that the product of AUXOSCCLK and the multiplier is between 110 and 550 MHz.
 - (c) Wait for the PLL to lock by polling the AUXPLLSTS.LOCKS bit. This will take 16 μ s plus 1024 AUXOSCCLK cycles.
 - (d) Repeat steps b and c to guarantee correct PLL lock.
 - (e) Connect the auxiliary PLL output clock (AUXPLLRAWCLK) to AUXPLLCLK by writing a 1 to AUXPLLCTL1.PLLCLKEN.

The auxiliary clock configuration can be changed at run time. Changing the AUXOSCCLK 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.

NOTE: If the AUXOSCCLK source is changed on the same AUXOSCCLK cycle as the multiplier, the PLL will be disabled but the AUXPLLMULT register will show the written value. This can happen when the system PLL is enabled before configuring the auxiliary PLL (CPUCLK >> AUXOSCCLK). To avoid this issue, wait two AUXOSCCLK cycles between changing the clock source and writing to AUXPLLMULT.

2.7.6.4 Clock Configuration Examples

Example 1: Using INTOSC2 (10 MHz) as a reference, generate a CPU frequency of 200 MHz - 3%:

```
CLKSRCCTL1.OSCCLKSRCSEL = 0x0
SYSPLLMULT.IMULT = 38 (0x26)
SYSPLLMULT.FMULT = .75 (0x3)
SYSCLKDIVSEL.PLLSYSCLKDIV = 2 (0x1)
SYSPLLCTL1.PLLCLKEN = 1
```

This gives a PLLRAWCLK of 387.5 MHz, well within the acceptable range of 110 - 550 MHz. The CPU frequency is 193.75 MHz, which is 200 MHz - 3.125%.

Example 2: Using a crystal (15 MHz) as a reference, generate a CPU frequency of 100 MHz and a USB clock of 60 MHz:

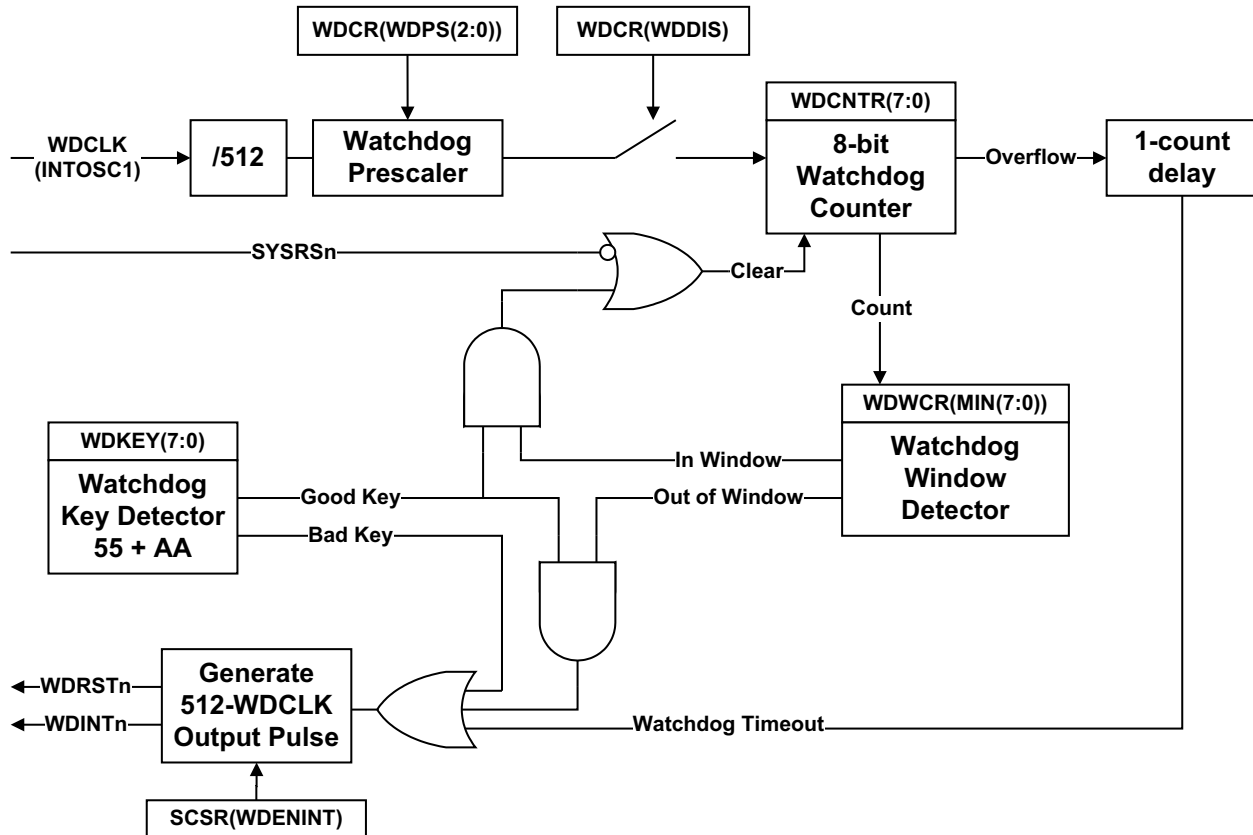
```
CLKSRCCTL1.OSCCLKSRCSEL = 0x1
SYSPLLMULT.IMULT = 26 (0x1A)
SYSPLLMULT.FMULT = .50 (0x2)
SYSCLKDIVSEL.PLLSYSCLKDIV = 4 (0x2)
SYSPLLCTL1.PLLCLKEN = 1
PERCLKDIVSEL.EPWMCLKDIV = 1 (0x0)
PERCLKDIVSEL.EMIF1CLKDIV = 1 (0x0)
PERCLKDIVSEL.EMIF2CLKDIV = 1 (0x0)
CLKSRCCTL2.AUXOSCCLKSRCSEL = 0x1
AUXPLLMULT.IMULT = 8 (0x08)
AUXPLLMULT.FMULT = .00 (0x0)
AUXCLKDIVSEL.AUXPLLDIV = 2 (0x1)
AUXPLLCTL1.PLLCLKEN = 1
```

This gives a PLLRAWCLK of 397.5 MHz and an AUXPLLRAWCLK of 120 MHz, both of which are in the acceptable range. The CPU frequency is 99.375 MHz. Crystals have tight frequency tolerances, which should keep the system clock from exceeding 100 MHz. The USB frequency is exactly 60 MHz. Since the CPU frequency is less than 100 MHz, the ePWM and EMIF clock dividers can be set to /1.

2.9 Watchdog Timers

The watchdog module generates an output pulse 512 watchdog clocks (WDCLKs) wide whenever the 8-bit watchdog up counter has reached its maximum value. The watchdog clock source is INTOSC1. Software must periodically write a 0x55 + 0xAA sequence into the watchdog key register to reset the watchdog counter. The counter can also be disabled. Figure 2-12 shows the various functional blocks within the watchdog module.

Figure 2-12. CPU Watchdog Module



2.9.1 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 2-8. 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.

Table 2-8. Example Watchdog Key Sequences (continued)

Step	Value Written to WDKEY	Result
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.

Step 3 in [Table 2-8](#) is the first action that enables the WDCNTR to be reset. 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.

2.9.2 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.

2.9.3 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.

Note: After a CPU1 watchdog reset, the boot ROMs will clear all of the system and message RAMs on both CPUs. After a CPU2 watchdog reset, CPU2's boot ROM will clear all of the CPU2 system and message RAMs.

- **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 $\overline{\text{WDINT}}$.

2.9.4 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 WAKEINT interrupt in the PIE during IDLE mode. User software must determine which peripheral caused the interrupt.

In STANDBY mode, all of the clocks to the peripherals are turned off within the CPU subsystem. The only peripheral that remains functional is the watchdog since the watchdog module runs off the oscillator clock (OSCCLK). The $\overline{\text{WDINT}}$ signal is fed to the Low Power Modes (LPM) block so that it can be used to wake the CPU from STANDBY low power mode. This feature is enabled by setting LPMCR.WDINTE = 1. See [Section 2.10](#) for details.

Note: If the watchdog interrupt is used to wake-up from an IDLE or STANDBY low power mode condition, software must make sure that the $\overline{\text{WDINT}}$ signal goes back high before attempting to reenter the IDLE or STANDBY 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 CPU1 watchdog 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.

2.9.5 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.

2.10 Low Power Modes

This device has three clock-gating, low-power modes and a special power-gating mode. 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* ([SPRU430](#)).

Low-power modes should not be entered into while a flash program or erase is ongoing.

The application should verify the following before entering STANDBY or HALT Mode:

1. Check the value of the GPIODAT register of the pin selected for STANDBY or HALT wake-up (GPIOLPMSEL0/1) prior to entering the Low-Power mode to ensure that the wake event has not already been asserted.
2. The LPMCR.QUALSTDBY register should be set to a value greater than the ratio of INTOSC1/PLLSYSCLK to ensure proper wake up. This is applicable to STANDBY only.

2.10.1 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. When one CPU is in IDLE, there is no effect on the other CPU subsystem.

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.

2.10.2 STANDBY

STANDBY is a more aggressive low-power mode that gates both the CPU clock and any peripheral clocks derived from the CPU's SYSCLK. The watchdog however, is left active. Like IDLE, this mode affects only one CPU subsystem. The other CPU subsystem and all of its peripherals are unaffected. STANDBY is best suited for an application where the wake-up signal will come from an external system (or CPU subsystem) rather than a peripheral input.

IPC interrupt 1 (flag 0), an NMI fired to the other CPU, or (optionally) a watchdog interrupt, will wake the CPU subsystem up from STANDBY mode. Any of GPIO0-63 can also be configured to wake up the subsystem when they are driven active low. Upon wakeup, the CPU receives a WAKEINT interrupt, even if it was woken by an IPCINT1 signal.

To enter STANDBY mode:

1. Set LPMCR.LPM to 0x1.
2. Enable the WAKEINT interrupt in the PIE.
3. For watchdog interrupt wakeup, set LPMCR.WDINTE to 1 and configure the watchdog to generate interrupts.
4. For GPIO wakeup, set GPIOLPMSEL0 and GPIOLPMSEL1 to connect the chosen GPIOs to the LPM module, and set LPMCR.QUALSTDBY to select the number of OSCCLK cycles for input qualification.
5. Execute the IDLE instruction to enter STANDBY.

To wake up from Standby mode:

1. Configure the desired GPIO to trigger the wakeup.
2. Drive the selected GPIO signal low; it must remain low for the number of OSCCLK cycles specified in the QUALSTDBY bits in the LPMCR register. If the signal is sampled high during this period, the count restarts.

At the end of the qualification period, the PLL enables the CLKIN to the CPU and the WAKEINT interrupt is latched in the PIE block. The WAKEINT interrupt can also be triggered by IPCINT1 sent from the other CPU and a watchdog interrupt.

The CPU is now out of STANDBY mode and can resume normal execution.

2.10.3 HALT

HALT is a global low-power mode that gates almost all system clocks and allows for power-down of oscillators and analog blocks. This mode affects both CPU subsystems. HALT can be used for additional power savings over putting both CPU subsystems in STANDBY, although the options for wakeup are more limited.

Similar to STANDBY, any of GPIO0-63 can be configured to wake up the system from HALT. No other wakeup option is available. However, CPU1's watchdog may still be clocked, and can be configured to produce a watchdog reset if a timeout mechanism is needed. On wakeup, both CPUs receive a WAKEINT interrupt.

To enter HALT mode:

1. Enable the WAKEINT interrupt in the PIE on both CPUs.
2. Put CPU2 into IDLE mode. (Using STANDBY will cause a duplicate WAKEINT on CPU2). CPU1 should verify this by checking the LPMSTAT register.
3. Set LPMCR.LPM to 0x2. Set GPIO_LPMSEL0 and GPIO_LPMSEL1 to connect the chosen GPIOs to the LPM module.
4. Set CLKSRCCTL1.WDHALTI to 1 to keep the CPU1 watchdog active and INTOSC1 and INTOSC2 powered up in HALT.
5. Set CLKSRCCTL1.WDHALTI to 0 to disable the CPU1 watchdog and power down INTOSC1 and INTOSC2 in HALT.
6. Execute the IDLE instruction on CPU1 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 CPU1.WAKEINT and CPU2.WAKEINT PIE interrupts.
2. Drive the wake-up GPIO high again to initiate the powering up of the SYSPLL and AUXPLL.
3. Wait 16us plus 1024 OSCCLK 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.

2.10.4 HIB

Hibernate (HIB) is a global low-power mode that gates the supply voltages to most of the system. This mode affects both CPU subsystems. HIB is essentially a controlled power-down with remote wakeup capability, and can be used to save power during long periods of inactivity. Because gating the supply voltage corrupts the state of the logic, a reset is required to exit HIB. To prevent external systems from being affected by the reset, HIB provides isolation of the I/O pin states as well as low-power data retention via the M0 and M1 memories.

Unlike the clock-gating modes, HIB does not have a true wakeup. Instead, GPIO41 becomes $\overline{\text{HIBWAKE}}$, an asynchronous reset signal. When the boot ROM detects a HIB wakeup, it will avoid clearing M0 and M1 and call a user-specified I/O restore function. To prevent glitches on internal and external signals, XRS will also generate a $\overline{\text{HIBWAKE}}$ signal during HIB. The I/O restore function should set up the GPIO control registers to match their pre-HIB state, then write a 1 to LPMCR.IOISODIS to deactivate I/O isolation. If the restore function does not disable isolation, the boot ROM will do it.

To enter HIB mode:

1. Save any necessary state to the M0 and M1 memories of both CPUs.
2. Put all I/Os in the desired state for isolation and deactivate any analog modules in use.
3. Write the address of the I/O restore function for each CPU to its IORESTOREADDR register.
4. Put CPU2 in reset, IDLE, or STANDBY.

5. Bypass the PLL by setting PLLCLKEN to 0.
6. Set CPU1's LPMCR.LPM to 0x3 and execute the IDLE instruction.

Any debugger connection will be lost on HIB entry since the JTAG logic is powered down.

Due to the loss of system state on HIB entry, it is possible for error information to be lost if an NMI is triggered while the IDLE instruction is in the pipeline. The ERRORSTS pin will be set and remain set until I/O isolation is disabled, but there will be no way to tell what caused the error.

To wake the device from HIB mode:

1. Assert the dedicated $\overline{\text{GPIOHIBWAKE}}$ pin (GPIO41) low to enable the power-up of the device clock sources.
2. Assert $\overline{\text{GPIOHIBWAKE}}$ pin high again. This triggers the power-up of the rest of the device.
3. Boot ROM code will execute on HIB wake-up. Boot ROM will read CPU1.RESC.HIBRESTn bit to determine this is a wake-up from HIB.
4. Boot ROM calls the I/O context restore routine. This I/O restore function should reconfigure the I/O configuration and do any other necessary application setup.

Since waking up from HIB mode is a type of reset, the device will enter the main function. The device is now out of HIB mode and can normal execution.

NOTE: The bootROM uses locations 0x02-0x122 on CPU1's M0 RAM and locations 0x02-0x80 on CPU2's M0 RAM. To prevent losing any data during HIB wake-up, avoid saving any critical data to these locations.

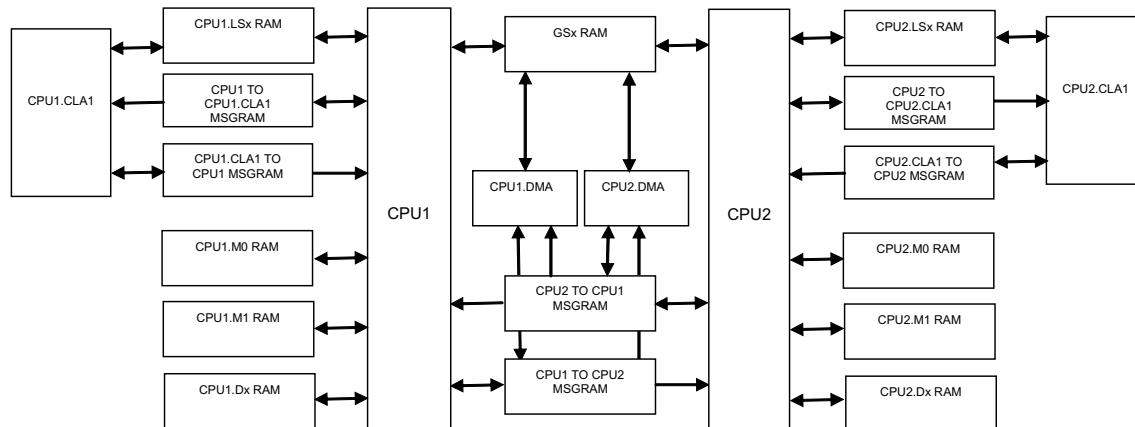
NOTE: The application must bypass the PLL before executing the IDLE instruction to enter HIB. If the PLL is not bypassed when entering HIB, there will be a brief current spike on the Vdd supply that may cause the device to reset.

2.11 Memory Controller Module

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

- dedicated to each CPU (M0, M1, and Dx RAMs),
- shared between the CPU and its own CLA (LSx RAM),
- shared between the CPU and DMA of both subsystems (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. There are also RAMs - called IPC MSGRAMs - that are used for interprocessor communication. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs, as well as IPC MSGRAMs, are enabled with the PARITY (both data and address) feature. Some of the dedicated memories are secure memory as well. Refer to [Section 2.13](#) 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 2-13](#) shows the configuration of these RAMs.

Figure 2-13. Memory Architecture


NOTE: All RAMs on these devices are SRAMs.

2.11.1 Functional Description

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

2.11.1.1 Dedicated RAM (Dx RAM)

Each CPU subsystem has four dedicated RAM blocks: M0, M1, D0, and D1. M0/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. All dedicated memories (except for M0/M1) are secure memory and also have the access protection (CPU write protection/CPU fetch protection) feature. Each type of access protection for each RAM block can be enabled/disabled by configuring the specific bit in the access protection register, allocated to each subsystem (DxACCPROT).

2.11.1.2 Local Shared RAM (LSx RAM)

RAM blocks which are dedicated to each subsystem and are accessible to its 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, mapped to each CPU subsystem. [Table 2-9](#) shows the LSx RAM features.

Table 2-9. 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	LSx memory is shared between CPU and CLA1

Table 2-9. Local Shared RAM (continued)

MSEL_LSx	CLAPGM_LSx	CPUx Allowed Access	CPUx.CLA1 Allowed Access	Comment
01	1	Emulation Read Emulation Write	Fetch Only	LSx memory is CLA1 program memory

2.11.1.3 Global Shared RAM (GSx RAM)

RAM blocks which are accessible from both the CPU and their respective DMA are called global shared RAMs (GSx RAMs). Each shared RAM can be owned by either CPU subsystem based on the configuration of their respective bits (one bit for each GSx memory) in the GSxMSEL register. When a particular GSx RAM block is owned by the CPU1 subsystem, CPU1 and CPU1.DMA have full access to that RAM block, whereas CPU2 and CPU2.DMA have only read access to it (no fetch/write access). Similarly, when a particular GSx RAM block is owned by the CPU2 subsystem, CPU1 and CPU1.DMA will have only read access (no fetch/write access) to that RAM block, whereas CPU2 and CPU2.DMA will have full access to it. [Table 2-10](#) shows the features of the GSx RAM.

Table 2-10. Global Shared RAM

GSxMSEL	CPU1	CPU1	CPU1	CPU1.DMA	CPU1.DMA	CPU2	CPU2	CPU2	CPU2.DMA	CPU2.DMA
	Fetch	Read	Write	Read	Write	Fetch	Read	Write	Read	Write
0	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No
1	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes

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

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 unless unlocked. 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 configuration is committed for a particular GSx RAM block, it can not be changed further until CPUx.SYSRS is issued. Only the CPU1 SW can change the master select configuration by writing into the GSxMSEL register, mapped on the CPU1. The GSxMSEL register, which is mapped to the CPU2 subsystem, is a status register which can only be used by CPU2 SW to know the master ownership for each GSx RAM block.

2.11.1.4 CPU Message RAM (CPU MSG RAM)

These RAM blocks can be used to share data between CPU1 and CPU2. Since these RAMs are used for interprocessor communication, they are also called IPC RAMs. The CPU MSGRAMs have CPU/DMA read/write access from its own CPU subsystem, and CPU/DMA read only access from the other subsystem.

This RAM has parity.

2.11.1.5 CLA Message RAM (CLA MSGRAM)

These RAM blocks are be 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.

2.11.1.6 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. Accesses from the same masters are arbitrated in a fixed priority manner, but the accesses from different masters are arbitrated using the round robin scheme.

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 2-14 represents the arbitration scheme on global shared memories:

Figure 2-14. Arbitration Scheme on Global Shared Memories

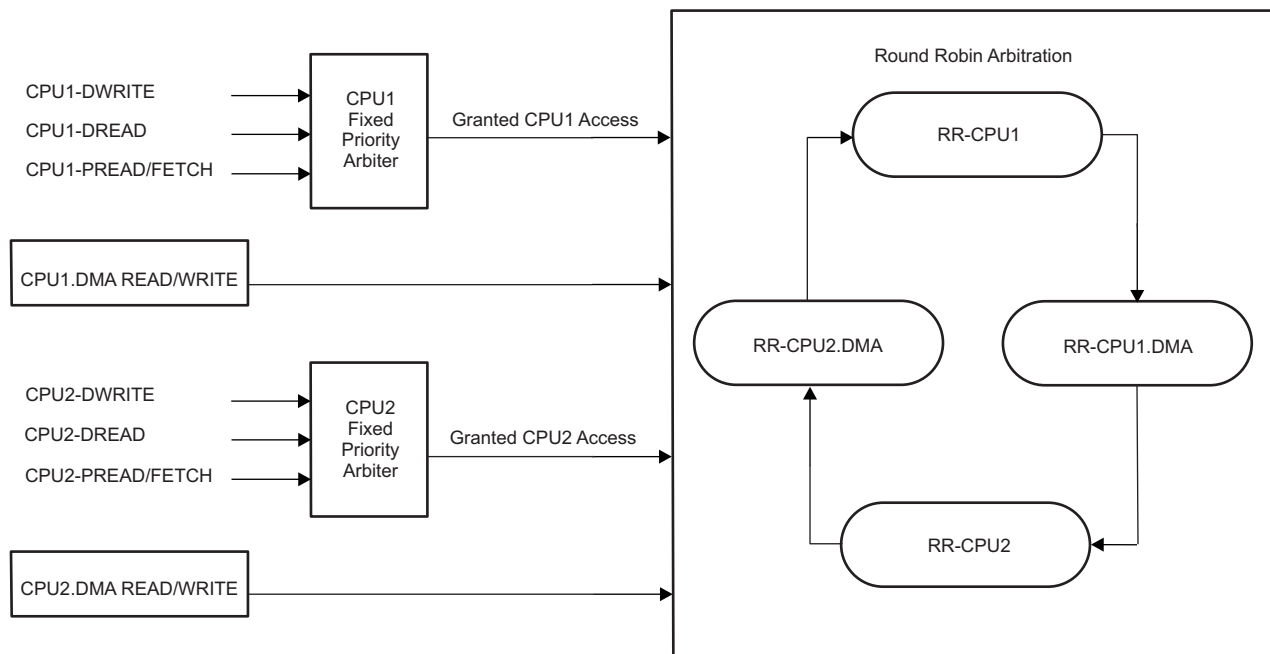
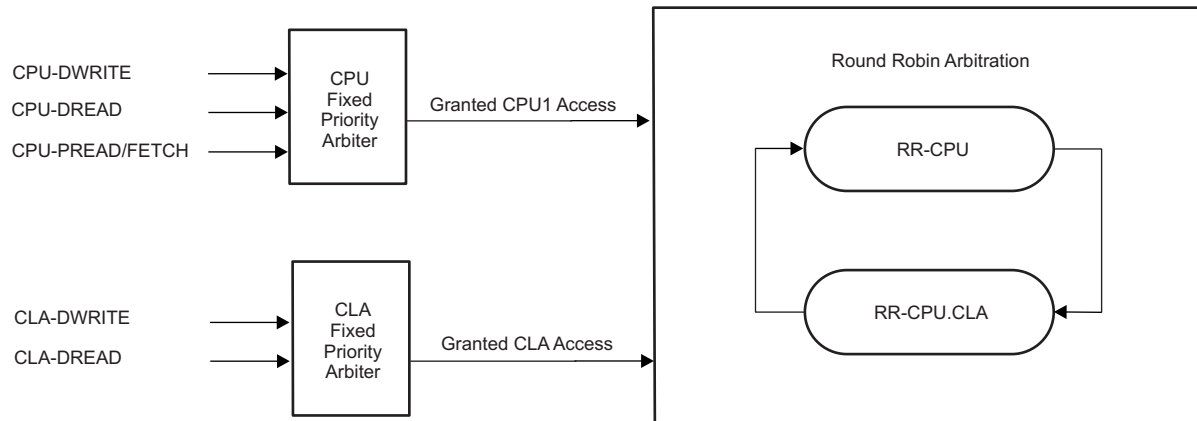


Figure 2-15 represents arbitration scheme on local shared memories.

Figure 2-15. Arbitration Scheme on Local Shared Memories


2.11.1.7 Access Protection

All RAM blocks except for M0/M1 on both subsystems 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.

2.11.1.7.1 CPU Fetch Protection

A CPU has execution permission from a memory, only if that memory is dedicated to that CPU or its respective subsystem is master for that memory (in case of GSx memory). When fetch accesses are allowed based on the mastership, it can be further 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.

There are two types of fetch protection violations:

- Non-master CPU fetch protection violation
- Master CPU fetch protection violation

If a fetch access is made to a memory by a non-master CPU, it is called a non-master fetch protection violation. If a fetch access is made to a dedicated or shared memory by the master CPU, and FETCHPROTx is set to '1' for that memory, the violation is called a master CPU fetch protection violation.

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

2.11.1.7.2 CPU Write Protection

A CPU has write permission to a memory only if that memory is dedicated to that CPU, or if its respective subsystem is the master for that memory (in case of GSx memory). When write accesses are allowed based on the mastership, it can be further 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.

There are two types of CPU write protection violations:

- Non-master CPU write protection violation
- Master CPU write protection violation

If a write access is made to a dedicated or shared memory by the master CPU and CPUWRPROTx is set to '1' for that memory, it's called a master CPU write protection violation.

If a write protection violation occurs, write gets ignored, a flag gets set into 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.

2.11.1.7.3 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 access from the CPU, including a read, will be blocked and the violation will be considered as a non-master access violation.

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

2.11.1.7.4 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 into the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched into 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.

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

2.11.1.7.6 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 into the appropriate access violation flag register, and the memory address for which the access violation occurred, gets latched into 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.

2.11.1.7.7 DMA Write Protection

The CPU1 or CPU2 DMA has write permission to a GSx memory only if its respective subsystem is master for that memory. When write accesses from a DMA are allowed based on the mastership, it can be further protected by setting the DMAWRPROTx bit of a specific register to '1.' If write access is done by the DMA to protected memory, a write protection violation occurs.

There are two types of DMA write protection violations:

- Non-master DMA write protection violation (only applicable to Sx memories)
- Master DMA write protection violation

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.

If a write protection violation occurs on CPU1, write is ignored and a DMAERR interrupt gets generated, whereas in the case of CPU2, a write is ignored and an access violation interrupt is generated if enabled in the interrupt enable register. 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 access from the CPU (including read) and data access from the CLA will be blocked, and violation will be considered as a non-master access violation. If the memory is configured as dedicated to the CPU, all access from the CLA will be blocked and the violation will be considered a non-master access violation.

2.11.1.8 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 then 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 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.

2.11.1.8.1 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 error 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 double-bit error due to another single-bit error at the same memory address.

2.11.1.8.2 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 into the correctable error threshold register, an interrupt is generated to the respective CPU, that is, 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 the master-specific status register and a flag gets set. Each of these registers are dedicated for each CPU subsystem.

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

[Table 2-11](#) summarizes different error situations that can arise. These need to be handled appropriately in the software, using the status and interrupt indications provided.

Table 2-11. 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 - CPUx/CPUx.DMA/CPUx.CLA1 CPU/DMA/CLA Read Error Address Register Data returned to CPUx/CPUx.DMA/CPUx.CLA1 is incorrect	NMI for CPUx access NMI for CPUx.DMA access NMI to CPU for CPUx.CLA1 access
Reads	Data read from memory	Single-bit error for ECC RAMs	Yes - CPUx/CPUx.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 - CPUx/CPUx.DMA/CPUx.CLA1 CPU/DMA/CLA Read Address Error Register Data returned to CPUx/CPUx.DMA/CPUx.CLA1 is incorrect	NMI to CPU for CPUx access NMI to CPU for CPUx.DMA access NMI to CPU for CPUx.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.

2.11.1.9 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 2-12. 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 2-13. 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

2.11.1.10 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.

In the case of GSx memory, only the CPU of the subsystem that is configured as the master for the particular GSx RAM block can initiate the RAM initialization.

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.

2.12 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 SECDDED safety feature.

2.12.1 Features

Features of flash memory include:

- Dedicated flash bank in the CPU1 subsystem (refer to the device data manual for the size of flash bank)
- Dedicated flash bank in the CPU2 subsystem (refer to the device data manual for the size of flash bank)
- Dedicated flash module controller (FMC) in the CPU1 and CPU2 subsystems for each bank
- 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 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)
- Single-flash pump shared by the CPU1 and CPU2 subsystems
- Hardware flash pump semaphore to control ownership of the pump between the two FMCs.
- Enhanced performance using the code-prefetch mechanism and data cache in CPU1-FMC and CPU2-FMC
- Configurable wait states to give the best performance for a given execution speed
- Safety Features
 - SECDDED-single error correction and double error detection is supported in both FMCs
 - 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 both FMCs
 - Simple flash API algorithms
 - Fast erase and program times (refer to the device data manual for details)
- Code Security Module (CSM) to prevent access to the flash by unauthorized persons (refer to [Section 2.13](#) for details)

2.12.2 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 *C2000 F021 Flash Application Programming Interface (API) Reference Guide* ([SPNU595](#)) 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* ([SPRU513](#)) for more information.

2.12.3 Default Flash Configuration

The following are flash module configuration settings at power-up in both CPU1 and CPU2 subsystems:

- Dedicated flash banks are in sleep power mode
- Shared pump is in sleep mode
- ECC is enabled
- Wait-states are set to the maximum (0xF)
- Code-prefetch mechanism and data cache are disabled in both FMCs

During the boot process, the boot ROM performs a dummy read of the Code Security Module (CSM) password locations in the OTP. This read is performed to unlock a new (or erased) device that has no password stored in it, so that flash programming or loading of code into CSM-protected SARAM can be performed. On devices with a password, this read has no effect and the device remains locked. One effect of this read is that the flash will transition from the sleep (reset) state to the active state.

User application software must initialize wait-states using the FRDCNTL register, and configure cache/prefetch features using the RD_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.

2.12.4 Flash Bank, OTP and Pump

There is a dedicated flash bank in the CPU1 subsystem called the CPU1 flash bank and a dedicated flash bank in the CPU2 subsystem called the CPU2 flash bank. Also, there is a one-time programmable (OTP) memory on the CPU1 subsystem 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.

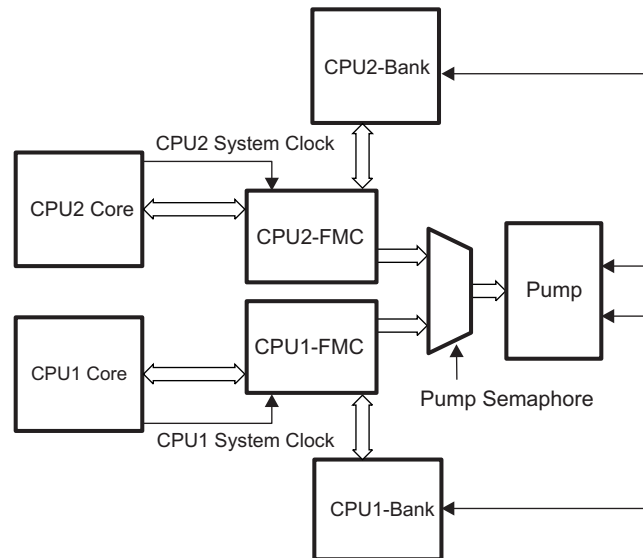
Both the CPU1 subsystem and CPU2 subsystem have a TI-OTP which contains manufacturing information like settings used by the flash state machine for erase and program operations, and so on. Users may read TI-OTP but it cannot be programmed or erased. For memory map and size information of the CPU1-Bank, CPU1 TI-OTP, CPU1 USER OTP, CPU2 flash bank, CPU2 TI-OTP, CPU2 USER-OTP and corresponding ECC locations, please refer to the device data manual.

The CPU1 flash bank/USER OTP and CPU2 flash bank/USER OTP share a common flash pump. A hardware semaphore, called the flash pump semaphore, is provided to control the access of the flash pump between the CPU1 subsystem and CPU2 subsystem.

Figure 2-20 depicts the user-programmable OTP locations in CPU1 USER-OTP and CPU2 USER-OTP. For more information on the functionality of these fields, please refer to [Section 2.13](#) and the *ROM Code and Peripheral Booting* chapter.

2.12.5 Flash Module Controller (FMC)

There is a dedicated flash module controller in both the CPU1 subsystem (CPU1-FMC) and the CPU2 subsystem (CPU2-FMC). The CPU1 in the CPU1 subsystem interfaces with the CPU1 flash module controller (CPU1-FMC), which in turn, interfaces with the CPU1 flash bank and shared pump to perform erase/program operations as well as to read data/execute code from the CPU1 flash bank.

Figure 2-16. FMC Interface with Core, Bank and Pump


The CPU2 in the CPU2 subsystem interfaces with the CPU2 flash module controller (CPU2-FMC) which in turn, interfaces with the CPU2 flash bank and shared pump to perform erase/program operations as well as to read data/execute code from the CPU2 flash bank. Control signals to the flash pump will be controlled by either CPU2-FMC or CPU1-FMC, depending on who gains the flash pump semaphore.

There is a state machine in both CPU1-FMC and CPU2-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 (refer to the *C2000 F021 Application Programming Interface (API) Reference Guide* [SPNU595](#), for details on Flash API).

[Section 2.12.6](#) through [Section 2.12.10](#) describe FMC in detail.

2.12.6 Flash and OTP Automatic Power-Down Modes

The flash bank and pump consume a significant amount of power when active. The flash module provides a mechanism to automatically power-down flash banks after they have not been accessed for some user programmable time. Special timers automatically sequence the power-up and power-down of the CPU1 flash bank and CPU2 flash bank independently of each other. The shared charge pump module has its own independent power-up and power-down timers as well.

2.12.6.1 Flash/OTP and Pump Power Modes and Wakeup

The flash bank 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 or Read 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. Also, if any bank is active or in standby mode, the charge pump is active, independent of the charge pump fallback power mode. 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 before putting the charge pump into active power mode. Refer to the register descriptions, [Section 2.14](#), for detailed information.

2.12.6.2 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 banks 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 banks and charge pump consume more power during flash startup and access.

The active grace periods (supported for CPU1 flash bank and CPU2 flash bank independently, 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 registers. The application software can program the fallback power mode to be standby or sleep mode to reduce power consumption, or program it to be active mode to keep the bank active regardless of counter settings (default). The charge pump AGP counter remains in its initialized state when any one of the banks is active, including the AGP counter of the bank. The charge pump AGP counter begins counting when both banks have become inactive.

As the charge pump is shared between CPU1-FMC and CPU2-FMC, the effective PAGP value and PMPPWR value used when powering down the pump will be of the FMC (out of CPU1-FMC and CPU2-FMC) which owns the pump. As the pump is shared between both CPU1 and CPU2 banks, if an access is made either to the CPU1-Bank or CPU2-Bank, the value of PMPPWR bit in both CPU1-FMC and CPU2-FMC changes to 1 (active). The application software can also check the current power mode of flash bank and charge pump by reading the FBPRDY register. Refer to the register descriptions, [Section 2.14](#), for detailed information.

2.12.7 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. The flash supports 0-wait accesses when the RWAIT bits are set to zero. This assumes that the CPU speed is low enough to accommodate the access time.

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 $\leq FCLK_{max}$, allowed maximum flash clock frequency with one wait state.

If RWAIT results in a fractional value when calculated using the above formula, RWAIT has to be rounded up to the nearest integer.

NOTE: Flash characterization is pending for these devices; therefore, the allowed maximum flash clock frequency with one wait state ($FCLK_{max}$) is not given in the device data manual. However from design simulations, $FCLK_{max}$ has been determined to be 50 MHz. This value will be updated later in the data manual when Flash characterization is complete.

2.12.8 Flash Read Interface

This section provides details about the data read modes to access flash bank/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.

2.12.8.1 FMC Flash Read Interface

2.12.8.1.1 Standard Read Mode

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

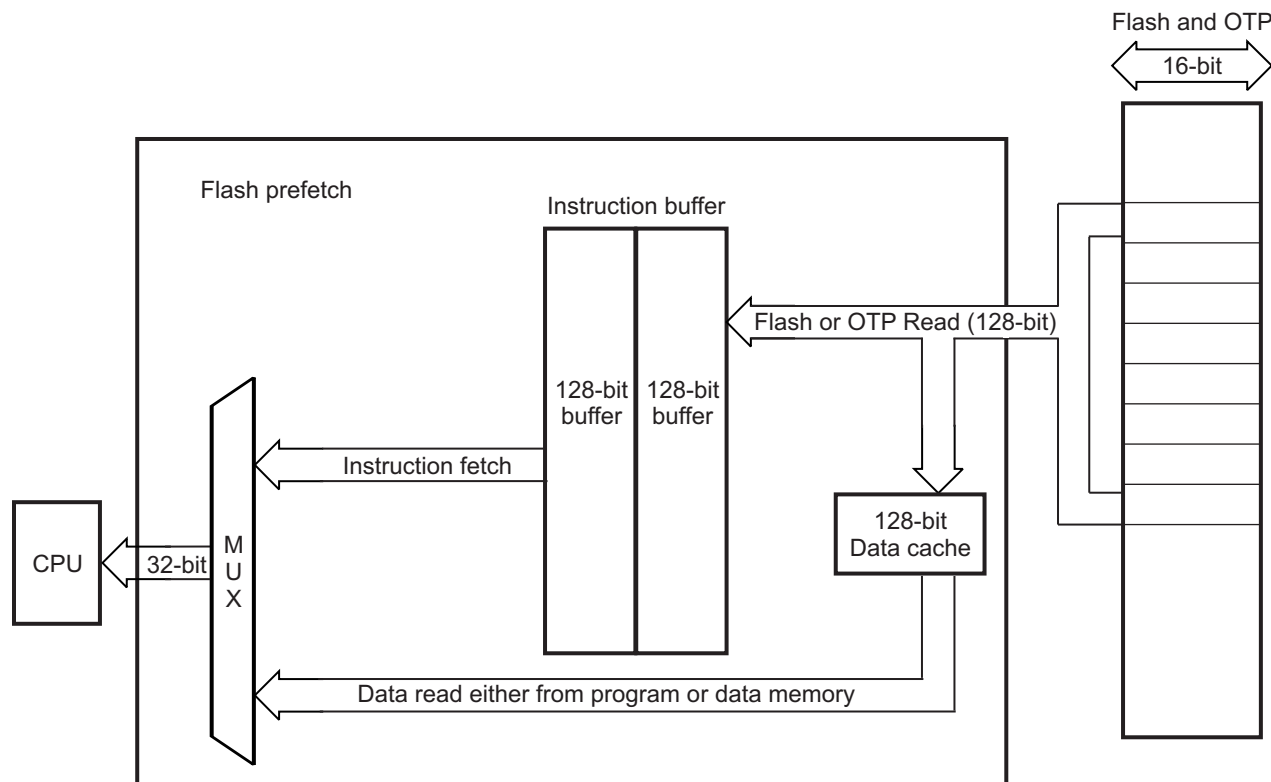
Prefetch buffers associated with prefetch mechanism and data cache are bypassed in standard read mode; therefore, every access to the flash/OTP is used by the CPU immediately, and every access creates a unique flash bank access.

Standard read 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 read 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 read mode (that is, maximum flash clock frequency with one wait-state - $FCLK_{max}$).

2.12.8.1.2 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 2-17](#) illustrates how this mode functions.

Figure 2-17. 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 fetch 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. If the prefetch mechanism is enabled, then the last row of 128 bits in the bank 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.

2.12.8.1.2.1 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.

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

- Reads of the 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, Zx-ECSLPSWD will return 0
 - Program space reads to Zx-CSMPSWD, Zx-ECSLPSWD will return 0
 - Program fetches to Zx-CSMPSWD, Zx-ECSLPSWD will return 0
- When the Code Security Module (CSM) 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.

2.12.9 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, refer to the *C2000 F021 Flash Application Programming Interface (API) Reference Guide* ([SPNU595](#)) for more information.

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

2.12.9.1 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. These commands are 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();
```

2.12.9.2 Program

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

This command is implemented by the following Flash API functions:

```
Fapi_issueProgrammingCommand();
```

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

2.12.9.3 Verify

After programming, the user must perform verify using API function Fapi_doVerify(). This function verifies the flash contents against supplied data in normal read mode and also in read-margin modes. Read-margin modes 0 and 1 are used to test cells for marginality. Read margin 0 is used to test cells for marginality of programmed cells. Read margin 1 is used to test cells for marginality of erased cells.

Application software is generally developed to enter read-margin mode during power-up and perform a full CRC check of the flash contents. In this way, a potential read mode failure can be identified prior to data loss or gain causing a bit flip.

NOTE: The read-margin modes are intended for diagnostic capabilities. Flash content is guaranteed for the lifetime specified in the datasheet.

Read margin modes should only be entered when executing from RAM. Banks must be powered up before entering a read margin mode. When entering a read margin mode, or changing from one read margin mode to the other, allow 1us delay before the first flash access to allow the flash banks to adjust to the new mode.

Read margin modes 0/1 are enabled by setting the RM0/RM1 bits in the special read control register FSPRD. The recommended procedures to enter, change and exit read margin mode are shown below.

To enter read-margin mode:

1. Start execution out of RAM.
2. Set the banks to remain powered up by writing 3 to bits 0:1 of FBFALLBACK register.
3. Turn on read margin 0 or 1 by writing 1 or 2 to the FSPRD register.
4. Do one dummy read from each bank to turn on the bank.
5. Flush the data cache by reading two flash locations separated by at least 32 bytes. The read data should be ignored for these reads.
6. Wait 1 us

Any reads will now be with the selected read margins. The application can now return to flash if desired.

To exit read-margin mode:

1. Start execution out of RAM.
2. Turn off read margin 0 or 1 by writing 0 to FSPRD register.
3. Flush the data cache by reading two flash locations separated by at least 32 bytes. The read data should be ignored for these reads.
4. Wait 1 us

Any reads will now be a standard read. The application can now return to flash if desired. The application can also set the FBFALLBACK register to the original values.

To change one read margin mode to the other:

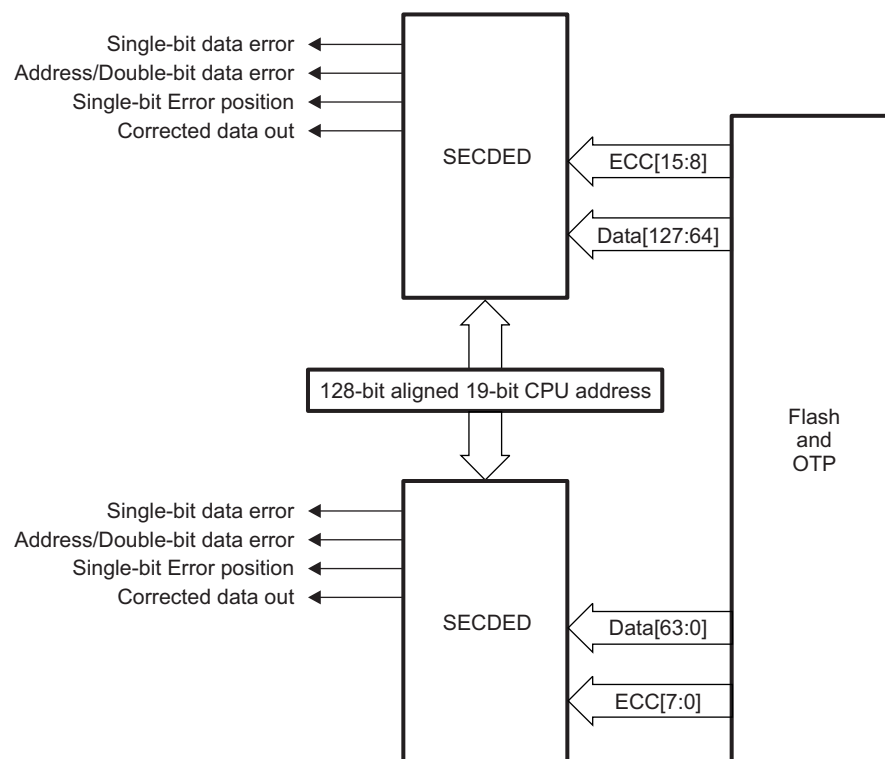
1. Follow the procedure to exit read margin mode above.
2. Follow the procedure to enter read margin mode above.

2.12.10 Error Correction Code (ECC) Protection

CPU1-FMC and CPU2-FMC contain an embedded single error correction and double error detection (SECEDED) module. SECEDED, when enabled, provides the capability to screen out memory faults. SECEDED 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. SECEDED 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 data. 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 the *TMS320C28x Assembly Language Tools User's Guide* ([SPRU513](#)) for more information on using the linker-supported ECC generation method.

Figure 2-18 illustrates the ECC logic inputs and outputs.

Figure 2-18. 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 SECEDED 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 SECEDED module to determine one of three conditions:

- No error occurred
- A correctable error (single bit data error) occurred

- A non-correctable 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 a non-correctable 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.

2.12.10.1 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.

2.12.10.2 Uncorrectable Error

Uncorrectable errors include address errors and double-bit errors in data/ECC. When SECDED 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.

2.12.10.3 SECDED Logic Correctness Check

Since error detection and correction logic are part of safety-critical logic, safety applications may need to ensure that the SECDED logic is always working properly always. For these safety concerns, in order to ensure the correctness of the SECDED logic, an ECC test mode is provided to test the correctness of ECC logic periodically. In this ECC test mode, instead of a data/ECC read from bank and address given to the FMC for read, a user-configurable 64-bit data/8-bit ECC check bits (FDATAH_TEST, FDATAL_TEST, FECC_TEST) and address (FADDR_TEST) are fed to SECDED logic. Using this test mode, users can introduce single-bit errors, double-bit errors, or address errors and check whether or not SECDED logic is catching those errors. Users can also check if SECDED 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 SECDED modules (out of the two SECDED 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 SECDED 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 FDATAx_TEST (FDATAL_TEST or FDATAH_TEST depending on which ECC logic the user wants to test) register
 - 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

2.12.10.4 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);

2.12.11 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 2.13](#), 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.

2.12.12 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.

2.13 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 access to on-chip secure memories and resources are blocked. The term “unsecure” implies 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™.

There are two CPUs on this device and each CPU subsystem has its own CSM for code protection. Each CPU subsystem’s CSM has dual-zone security, which means the CPU1 subsystem has two zones (zone1/zone2) and CPU2 also has two zones (zone1/zone2).

2.13.1 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).
- **CLA:** The CLA is a secure resource which can be allocated to either zone by configuring the GRABRAM location in the USER OTP. CLA configuration can only be performed by code running from the zone to which it has been allocated. The CLA message RAMs also belong to the same zone.
- **RAM:** All Dx and 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 USER OTP.
- **Flash Sectors:** Flash Sectors can be secure on this device. Each Flash sector can be allocated to either zone by configuring the respective GRABSECT location in the 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 2-14 shows the status of a RAM block based on the configuration in GRABRAM register.

Table 2-14. RAM Status

GRAM_RAMx Bits in Z1_GRABRAMR Register	GRAM_RAMx Bits in Z2_GRABRAMR Register	Ownership
00	XX	GRAM_RAMx is inaccessible
XX	00	GRAM_RAMx is inaccessible
Differential Value (01/10)	Differential Value (01/10)	GRAM_RAMx is inaccessible
Differential Value (01/10)	11	GRAM_RAMx belongs to Z1
11	Differential Value (01/10)	GRAM_RAMx belongs to Z2
11	11	GRAM_RAMx is Non-Secure

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 its dedicated OTP location based on a zone-specific link pointer. A zone can be unsecured by executing the password match flow (PMF), described in [Section 2.13.3.3.2](#).

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 2-15](#) shows the levels of security.

Table 2-15. 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	Non-Secure	Anywhere	Full access for CPU and JTAG port to secure memory of that zone.

If the password locations of a zone have all 128 bits as ones, the zone is considered unsecure. Since new Flash devices have erased Flash (all ones), only a read of the password locations is required to bring any zone into unsecure mode. If the password locations of a zone have all 128 bits as zeros, the zone is secure, regardless of the contents of the CSMKEY registers. This means the zone can't be unlocked using PMF, the password match flow described in [Section 2.13.3.3.2](#). Therefore, the user should never use all zeros as a password. A password of all zeros will prevent debug of secure code or reprogramming the Flash.

CSMKEY registers are user-accessible registers that are used to unsecure the zones.

2.13.1.1 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 the specific CPU subsystem for which the ECSL violation occurred. 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.

2.13.1.2 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.

2.13.1.3 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 sectors 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.

2.13.1.4 Password Lock

The password locations 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.

2.13.1.5 Link Pointer and Zone Select

For each of the two security zones of each CPU subsystem on this device, a dedicated OTP block exists that holds the configuration related to zone's security. The following are the available programmed configurations:

- Zx-LINKPOINTER1
- Zx-LINKPOINTER2
- Zx-LINKPOINTER3
- Zx-PSWDLOCK
- Zx-CRCLOCK
- Zx-BOOTCTRL
- Zx-EXEONLYRAM
- Zx-EXEONLYSECT
- Zx-GRABRAM
- Zx-GRABSECT
- Zx-CSMPASSWORD

Since OTP can't be erased, to provide flexibility of configuring some of the security settings like CSM passwords, allocation of RAM/Flash sectors and their attributes, multiple times by the user, the following configurations are placed in zone select regions of each zone's OTP Flash.

- Zx-EXEONLYRAM
- Zx-EXEONLYSECT
- Zx-GRABRAM
- Zx-GRABSECT
- Zx-CSMPASSWORD

The location of the zone select region in OTP is decided based on the value of three 29-bit link pointers (Zx-LINKPOINTERx) programmed in the OTP of each zone of both CPU subsystems. 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

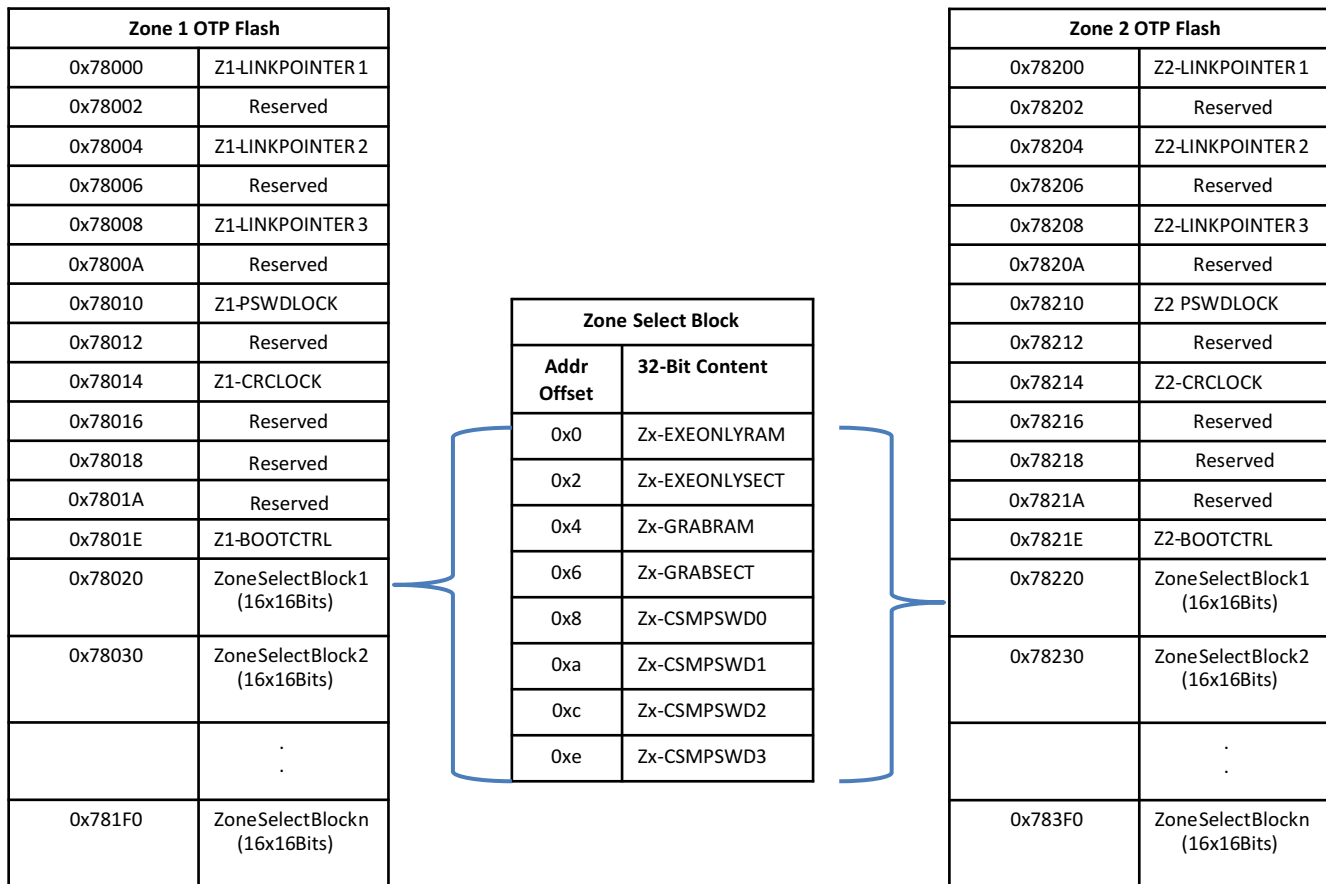
all the three values (bit-wise voting logic). Since in OTP, a '1' can be flipped by the user to '0' but '0' can't be flipped to '1' (no erase operation for OTP), the most significant bit position in the resolved link pointer which is '0', defines the valid base address for the zone select region. While generating the final link pointer value, if the bit patterns is not one of those listed in [Figure 2-19](#), 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 2-19. Storage of Zone-Select Bits in OTP

Zx-LINKPOINTER	Addr Offset Of Zone-Select Block
32'bxxx11111111111111111111111111111111	0x20
32'bxxx11111111111111111111111111111111 0	0x30
32'bxxx11111111111111111111111111111111 0x	0x40
32'bxxx11111111111111111111111111111111 0xx	0x50
32'bxxx11111111111111111111111111111111 0xxx	0x60
32'bxxx11111111111111111111111111111111 0xxxx	0x70
32'bxxx11111111111111111111111111111111 0xxxxx	0x80
32'bxxx11111111111111111111111111111111 0xxxxxx	0x90
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xa0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xb0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xc0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xd0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xe0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0xf0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x100
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x110
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x120
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x130
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x140
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x150
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x160
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x170
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x180
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x190
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1a0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1b0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1c0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1d0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1e0
32'bxxx11111111111111111111111111111111 0xxxxxxx	0x1f0

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

NOTE: Address locations for other security settings (PSWDLCK/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.

Figure 2-20. Location of Zone-Select Block Based on Link-Pointer


2.13.1.5.1 C Code Example to get Zone Select Block Addr for Zone1

```

unsigned long LinkPointer;
unsigned long *ZoneSelBlockPtr;
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 << 2;

while ((ZeroFound == 0) && (bitpos > -1))
{
    if ((LinkPointer & 0x80000000) == 0)
    {
        ZeroFound = 1;
        ZoneSelBlockPtr = (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;
}
```

2.13.1.6 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. Each zone has its own CSM passwords; read and write accesses are not allowed to resources owned by Z1 from code running from memory allocated to Z2 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. Similarly, to program the security settings in OTP Flash, the user must either unlock the CSM of the respective zone or execute the Flash programming code from secure memory which belongs to the same zone. the OTP content cannot be erased.

This device has only one Flash pump used for erase/program operation of normal Flash and OTP Flash. 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.

2.13.1.7 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.

2.13.1.8 SafeCRC

Since reads from EXEONLY memories are not allowed, the user cannot calculate the CRC for content in EXEONLY memories using the VCU-II. 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: Code Security Module 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.

2.13.2 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.

- Dummy Read to address location of SECDC (0x703F0, TI-reserved register) in TI OTP
- Dummy Read to address location of Z1_LINKPOINTER1 in Z1 OTP
- Dummy Read to address location of Z1_LINKPOINTER2 in Z1 OTP
- Dummy Read to address location of Z1_LINKPOINTER3 in Z1 OTP
- Dummy Read to address location of Z1_PSWDLOCK in Z1 OTP
- Dummy Read to address location of Z1_CRCLOCK in Z1 OTP
- Dummy Read to address location 0x78018 in Z1 OTP
- Dummy Read to address location of Z1_BOOTCTRL in Z1 OTP
- Read to memory map register of Z1_LINKPOINTER in DCSM module to calculate the address of zone select block for Z1
- Dummy read to address location of Z1_EXEONLYRAM in Z1 OTP
- Dummy read to address location of Z1_EXEONLYSECT in Z1 OTP
- Dummy read to address location of Z1_GRABRAM in Z1 OTP
- Dummy read to address location of Z1_GRABSECT in Z1 OTP
- Dummy Read to address location of Z2_LINKPOINTER1 in Z2 OTP
- Dummy Read to address location of Z2_LINKPOINTER2 in Z2 OTP
- Dummy Read to address location of Z2_LINKPOINTER3 in Z2 OTP
- Dummy Read to address location of Z2_PSWDLOCK in Z2 OTP
- Dummy Read to address location of Z2_CRCLOCK in Z2 OTP
- Dummy Read to address location 0x78218 in Z2 OTP
- Dummy Read to address location of Z2_BOOTCTRL in Z2 OTP
- Read to memory map register of Z2_LINKPOINTER in DCSM module to calculate the address of zone select block for Z2
- Dummy read to address location of Z2_EXEONLYRAM in Z2 OTP
- Dummy read to address location of Z2_EXEONLYSECT in Z2 OTP
- Dummy read to address location of Z2_GRABRAM in Z2 OTP
- Dummy read to address location of Z2_GRABSECT in Z2 OTP

2.13.3 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.

2.13.3.1 Environments That Require Security Unlocking

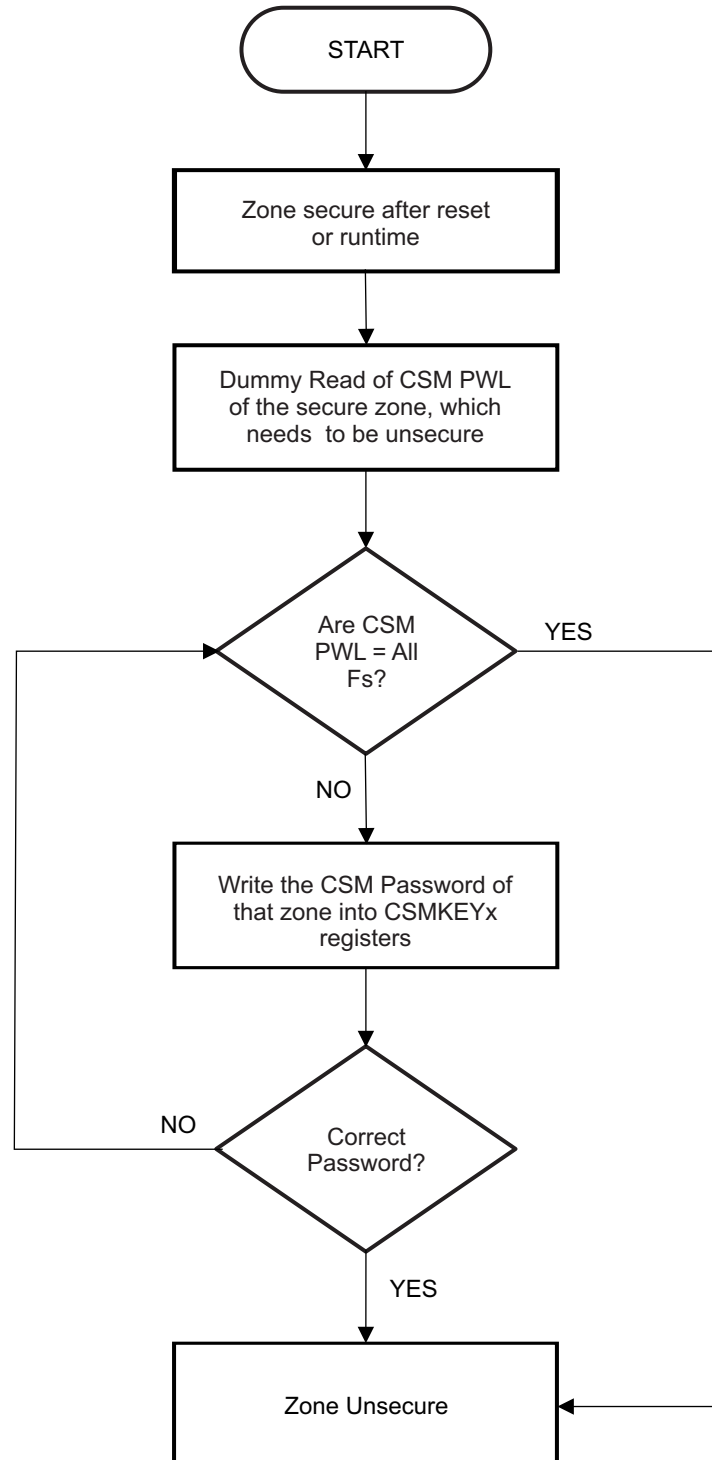
The following are the typical situations under which unsecuring can be required:

- 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 2-21](#) 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.

2.13.3.2 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 2-21](#) shows how PMF helps to initialize the security logic registers and disable security logic.

Figure 2-21. CSM Password Match Flow (PMF)


2.13.3.3 Unsecuring Considerations for Zones With and Without Code Security

Case 1 and Case 2 provide unsecuring considerations for zones with and without code security.

- **Case 1: Zone With Code Security**

A zone with code security should have a predetermined password stored in the password locations of that zone. In addition, reserved password locations of that zone should be programmed with all 0x0000

and not used for program and/or data storage. The following are steps to unsecure any secure zone:

1. Perform a dummy read of the password locations of that zone.
2. Write the password into the CSMKEY registers.
3. If the password is correct, the zone becomes unsecure; otherwise, it stays secure.

- **Case 2: Zone Without Code Security**

A zone without code security should have 0x FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF (128 bits of all ones) stored in the password locations. The following are steps to use this zone:

1. At reset, the CSM will lock memory regions protected by the CSM.
2. Perform a dummy read of the password locations.
3. Since the password is all ones, this alone will unlock the zone and all the secure memories dedicated to that zone are fully accessible immediately after this operation is completed.

NOTE: Even if a zone is not protected with a password (all password locations all ones), the CSM will lock at reset. Thus, a dummy read operation must still be performed on these zones prior to reading, writing, or programming secure memory if the code performing the access is executing from outside of the CSM protected memory region. The Boot ROM code does this dummy read for convenience.

2.13.3.3.1 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 sel 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
```

2.13.3.3.2 C Code Example to Resecure C28x Zone1

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

2.13.3.4 Environments That Require ECSL Unlocking

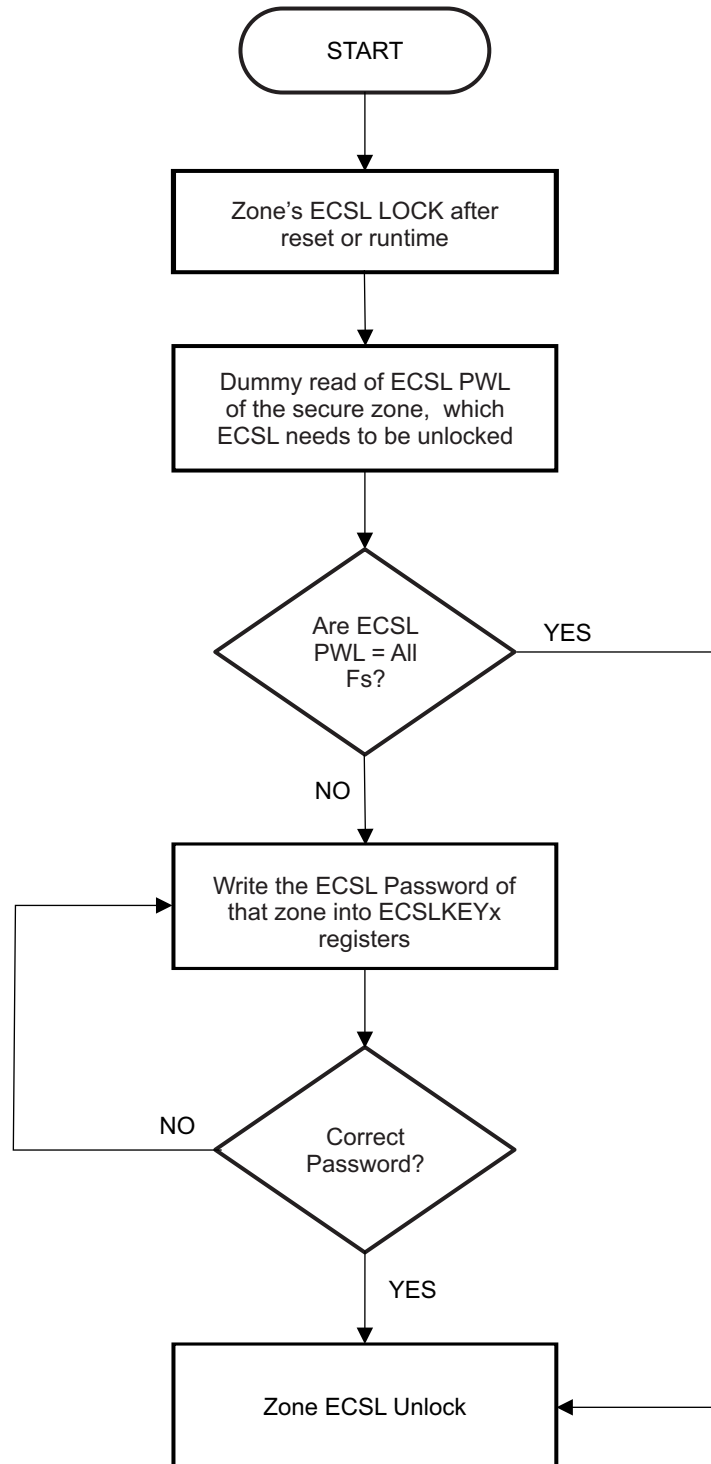
The following are the typical situations under which unsecuring can be required:

- 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).

2.13.3.5 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 2-22](#) shows how the PMF helps to initialize the security logic registers and disable security logic.

Figure 2-22. ECSL Password Match Flow (PMF)



2.13.3.6 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 CAM password locations of that Zone
- Write the password into the CSMKEYx registers, corresponding to that Zone.
- If the password is correct, the ECSL gets disabled; otherwise, it stays enabled.

2.13.3.6.1 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
```

2.14 Registers

2.14.1 Base Addresses

Table 2-16. 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_017F
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 ⁽¹⁾	DEV_CFG_REGS	0x0005_D000	0x0005_D17F
ClkCfgRegs	CLK_CFG_REGS	0x0005_D200	0x0005_D2FF
CpuSysRegs	CPU_SYS_REGS	0x0005_D300	0x0005_D3FF
RomPrefetchRegs ⁽¹⁾	ROM_PREFETCH_REGS	0x0005_E608	0x0005_E609
DcsmZ1Regs	DCSM_Z1_REGS	0x0005_F000	0x0005_F02F
DcsmZ2Regs	DCSM_Z2_REGS	0x0005_F040	0x0005_F05F
DcsmCommonRegs	DCSM_COMMON_REGS	0x0005_F070	0x0005_F07F
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 ⁽¹⁾	ROM_WAIT_STATE_REGS	0x0005_F540	0x0005_F541
Flash0CtrlRegs	FLASH_CTRL_REGS	0x0005_F800	0x0005_FAFF
Flash0EccRegs	FLASH_ECC_REGS	0x0005_FB00	0x0005_FB3F

⁽¹⁾ Only available on CPU1.

2.14.2 CPUTIMER_REGS Registers

Table 2-17 lists the memory-mapped registers for the CPUTIMER_REGS. All register offset addresses not listed in Table 2-17 should be considered as reserved locations and the register contents should not be modified.

Table 2-17. CPUTIMER_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	TIM	CPU-Timer, Counter Register		Go
2h	PRD	CPU-Timer, Period Register		Go
4h	TCR	CPU-Timer, Control Register		Go
6h	TPR	CPU-Timer, Prescale Register		Go
7h	TPRH	CPU-Timer, Prescale Register High		Go

2.14.2.1 TIM Register (Offset = 0h) [reset = FFFFh]

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

CPU-Timer, Counter Register

Figure 2-23. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-18. TIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	0h	<p>CPU-Timer Counter Registers</p> <p>The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDRH:TDDR+1) clock cycles, where TDDRH:TDDR is the timer prescale dividedown value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers. The timer interrupt (TINT) signal is generated.</p>
15-0	LSW	R/W	FFFFh	<p>CPU-Timer Counter Registers</p> <p>The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDRH:TDDR+1) clock cycles, where TDDRH:TDDR is the timer prescale dividedown value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers. The timer interrupt (TINT) signal is generated.</p>

2.14.2.2 PRD Register (Offset = 2h) [reset = 1FFFFh]

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

CPU-Timer, Period Register

Figure 2-24. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-19. PRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	1h	<p>CPU-Timer Counter Registers</p> <p>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).</p>
15-0	LSW	R/W	FFFFh	<p>CPU-Timer Counter Registers</p> <p>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).</p>

2.14.2.3 TCR Register (Offset = 4h) [reset = 1h]

TCR is shown in [Figure 2-25](#) and described in [Table 2-20](#).

CPU-Timer, Control Register

Figure 2-25. TCR Register

15	14	13	12	11	10	9	8
TIF	TIE	RESERVED		FREE	SOFT	RESERVED	
R/W1toClr-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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-20. TCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	TIF	R/W1toClr	0h	CPU-Timer Overflow Flag. 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. 0h (R/W) = The CPU-Timer has not decremented to zero. Writes of 0 are ignored. 1h (R/W) = This flag gets set when the CPU-timer decrements to zero. Writing a 1 to this bit clears the flag.
14	TIE	R/W	0h	CPU-Timer Interrupt Enable. 0h (R/W) = The CPU-Timer interrupt is disabled. 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.
13-12	RESERVED	R	0h	Reserved
11	FREE	R/W	0h	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. 0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop) 1h (R/W) = Stop after the TIMH:TIM decrements to 0 (soft stop) In the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition). 2h (R/W) = Free run 3h (R/W) = Free run
10	SOFT	R/W	0h	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.
9-6	RESERVED	R	0h	Reserved
5	TRB	R/W	0h	Timer reload 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).

Table 2-20. TCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	TSS	R/W	0h	<p>CPU-Timer stop status bit.</p> <p>TSS is a 1-bit flag that stops or starts the CPU-timer.</p> <p>0h (R/W) = Reads of 0 indicate the CPU-timer is running.</p> <p>To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the CPU-timer immediately starts.</p> <p>1h (R/W) = Reads of 1 indicate that the CPU-timer is stopped.</p> <p>To stop the CPU-timer, set TSS to 1.</p>
3-0	RESERVED	R	1h	Reserved

2.14.2.4 TPR Register (Offset = 6h) [reset = 0h]

TPR is shown in [Figure 2-26](#) and described in [Table 2-21](#).

CPU-Timer, Prescale Register

Figure 2-26. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-21. 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>
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>

2.14.2.5 TPRH Register (Offset = 7h) [reset = 0h]

TPRH is shown in [Figure 2-27](#) and described in [Table 2-22](#).

CPU-Timer, Prescale Register High

Figure 2-27. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-22. TPRH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	PSCH	R	0h	See description of TIMERxTPR.
7-0	TDDRH	R/W	0h	See description of TIMERxTPR.

2.14.3 PIE_CTRL_REGS Registers

Table 2-23 lists the memory-mapped registers for the PIE_CTRL_REGS. All register offset addresses not listed in Table 2-23 should be considered as reserved locations and the register contents should not be modified.

Table 2-23. PIE_CTRL_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	PIECTRL	Control Register		Go
1h	PIEACK	Acknowledge Register		Go
2h	PIEIER1	INT1 Group Enable Register		Go
3h	PIEIFR1	INT1 Group Flag Register		Go
4h	PIEIER2	INT2 Group Enable Register		Go
5h	PIEIFR2	INT2 Group Flag Register		Go
6h	PIEIER3	INT3 Group Enable Register		Go
7h	PIEIFR3	INT3 Group Flag Register		Go
8h	PIEIER4	INT4 Group Enable Register		Go
9h	PIEIFR4	INT4 Group Flag Register		Go
Ah	PIEIER5	INT5 Group Enable Register		Go
Bh	PIEIFR5	INT5 Group Flag Register		Go
Ch	PIEIER6	INT6 Group Enable Register		Go
Dh	PIEIFR6	INT6 Group Flag Register		Go
Eh	PIEIER7	INT7 Group Enable Register		Go
Fh	PIEIFR7	INT7 Group Flag Register		Go
10h	PIEIER8	INT8 Group Enable Register		Go
11h	PIEIFR8	INT8 Group Flag Register		Go
12h	PIEIER9	INT9 Group Enable Register		Go
13h	PIEIFR9	INT9 Group Flag Register		Go
14h	PIEIER10	INT10 Group Enable Register		Go
15h	PIEIFR10	INT10 Group Flag Register		Go
16h	PIEIER11	INT11 Group Enable Register		Go
17h	PIEIFR11	INT11 Group Flag Register		Go
18h	PIEIER12	INT12 Group Enable Register		Go
19h	PIEIFR12	INT12 Group Flag Register		Go

2.14.3.1 PIELCTRL Register (Offset = 0h) [reset = 0h]

PIECTRL is shown in [Figure 2-28](#) and described in [Table 2-24](#).

Control Register

Figure 2-28. PIELCTRL 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-24. PIELCTRL 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 is shown. The vector value can be read by the user to determine which interrupt generated the vector fetch.
0	ENPIE	R/W	0h	Enable vector fetching from ePIE block. When this bit is set to 1, all vectors are fetched from the ePIE vector table. If this bit is set to 0, the ePIE block is disabled and vectors are fetched as normal. All ePIE block registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the ePIE block is disabled. Note: ,The state of this bit will be reflected on the ENPIES signal, which is an output from the ePIE block.

2.14.3.2 PIEACK Register (Offset = 1h) [reset = 0h]

PIEACK is shown in [Figure 2-29](#) and described in [Table 2-25](#).

Acknowledge Register

Writing a 1 to the respective interrupt bit enables the ePIE block to drive a pulse into the core interrupts input, if an interrupt is pending on any of the group interrupts.

Reading this register indicates if an interrupt is pending in the respective group.

Note: Writes of 0 are ignored.

Figure 2-29. PIEACK Register

15	14	13	12	11	10	9	8
RESERVED				ACK12	ACK11	ACK10	ACK9
R=0-0h				R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h
7	6	5	4	3	2	1	0
ACK8	ACK7	ACK6	ACK5	ACK4	ACK3	ACK2	ACK1
R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h	R/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-25. PIEACK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R=0	0h	Reserved
11	ACK12	R/W=1	0h	Acknowledge PIE Interrupt Group 12
10	ACK11	R/W=1	0h	Acknowledge PIE Interrupt Group 11
9	ACK10	R/W=1	0h	Acknowledge PIE Interrupt Group 10
8	ACK9	R/W=1	0h	Acknowledge PIE Interrupt Group 9
7	ACK8	R/W=1	0h	Acknowledge PIE Interrupt Group 8
6	ACK7	R/W=1	0h	Acknowledge PIE Interrupt Group 7
5	ACK6	R/W=1	0h	Acknowledge PIE Interrupt Group 6
4	ACK5	R/W=1	0h	Acknowledge PIE Interrupt Group 5
3	ACK4	R/W=1	0h	Acknowledge PIE Interrupt Group 4
2	ACK3	R/W=1	0h	Acknowledge PIE Interrupt Group 3
1	ACK2	R/W=1	0h	Acknowledge PIE Interrupt Group 2
0	ACK1	R/W=1	0h	Acknowledge PIE Interrupt Group 1

2.14.3.3 PIEIER1 Register (Offset = 2h) [reset = 0h]

PIEIER1 is shown in [Figure 2-30](#) and described in [Table 2-26](#).

INT1 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-30. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-26. PIEIER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT1.16
14	INTx15	R/W	0h	Interrupt Enable for INT1.15
13	INTx14	R/W	0h	Interrupt Enable for INT1.14
12	INTx13	R/W	0h	Interrupt Enable for INT1.13
11	INTx12	R/W	0h	Interrupt Enable for INT1.12
10	INTx11	R/W	0h	Interrupt Enable for INT1.11
9	INTx10	R/W	0h	Interrupt Enable for INT1.10
8	INTx9	R/W	0h	Interrupt Enable for INT1.9
7	INTx8	R/W	0h	Interrupt Enable for INT1.8
6	INTx7	R/W	0h	Interrupt Enable for INT1.7
5	INTx6	R/W	0h	Interrupt Enable for INT1.6
4	INTx5	R/W	0h	Interrupt Enable for INT1.5
3	INTx4	R/W	0h	Interrupt Enable for INT1.4
2	INTx3	R/W	0h	Interrupt Enable for INT1.3
1	INTx2	R/W	0h	Interrupt Enable for INT1.2
0	INTx1	R/W	0h	Interrupt Enable for INT1.1

2.14.3.4 PIEIFR1 Register (Offset = 3h) [reset = 0h]

PIEIFR1 is shown in [Figure 2-31](#) and described in [Table 2-27](#).

INT1 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-31. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-27. PIEIFR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT1.16
14	INTx15	R/W	0h	Interrupt Flag for INT1.15
13	INTx14	R/W	0h	Interrupt Flag for INT1.14
12	INTx13	R/W	0h	Interrupt Flag for INT1.13
11	INTx12	R/W	0h	Interrupt Flag for INT1.12
10	INTx11	R/W	0h	Interrupt Flag for INT1.11
9	INTx10	R/W	0h	Interrupt Flag for INT1.10
8	INTx9	R/W	0h	Interrupt Flag for INT1.9
7	INTx8	R/W	0h	Interrupt Flag for INT1.8
6	INTx7	R/W	0h	Interrupt Flag for INT1.7
5	INTx6	R/W	0h	Interrupt Flag for INT1.6
4	INTx5	R/W	0h	Interrupt Flag for INT1.5
3	INTx4	R/W	0h	Interrupt Flag for INT1.4
2	INTx3	R/W	0h	Interrupt Flag for INT1.3
1	INTx2	R/W	0h	Interrupt Flag for INT1.2
0	INTx1	R/W	0h	Interrupt Flag for INT1.1

2.14.3.5 PIEIER2 Register (Offset = 4h) [reset = 0h]

PIEIER2 is shown in [Figure 2-32](#) and described in [Table 2-28](#).

INT2 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-32. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-28. PIEIER2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT2.16
14	INTx15	R/W	0h	Interrupt Enable for INT2.15
13	INTx14	R/W	0h	Interrupt Enable for INT2.14
12	INTx13	R/W	0h	Interrupt Enable for INT2.13
11	INTx12	R/W	0h	Interrupt Enable for INT2.12
10	INTx11	R/W	0h	Interrupt Enable for INT2.11
9	INTx10	R/W	0h	Interrupt Enable for INT2.10
8	INTx9	R/W	0h	Interrupt Enable for INT2.9
7	INTx8	R/W	0h	Interrupt Enable for INT2.8
6	INTx7	R/W	0h	Interrupt Enable for INT2.7
5	INTx6	R/W	0h	Interrupt Enable for INT2.6
4	INTx5	R/W	0h	Interrupt Enable for INT2.5
3	INTx4	R/W	0h	Interrupt Enable for INT2.4
2	INTx3	R/W	0h	Interrupt Enable for INT2.3
1	INTx2	R/W	0h	Interrupt Enable for INT2.2
0	INTx1	R/W	0h	Interrupt Enable for INT2.1

2.14.3.6 PIEFR2 Register (Offset = 5h) [reset = 0h]

PIEFR2 is shown in [Figure 2-33](#) and described in [Table 2-29](#).

INT2 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-33. PIEFR2 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-29. PIEFR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT2.16
14	INTx15	R/W	0h	Interrupt Flag for INT2.15
13	INTx14	R/W	0h	Interrupt Flag for INT2.14
12	INTx13	R/W	0h	Interrupt Flag for INT2.13
11	INTx12	R/W	0h	Interrupt Flag for INT2.12
10	INTx11	R/W	0h	Interrupt Flag for INT2.11
9	INTx10	R/W	0h	Interrupt Flag for INT2.10
8	INTx9	R/W	0h	Interrupt Flag for INT2.9
7	INTx8	R/W	0h	Interrupt Flag for INT2.8
6	INTx7	R/W	0h	Interrupt Flag for INT2.7
5	INTx6	R/W	0h	Interrupt Flag for INT2.6
4	INTx5	R/W	0h	Interrupt Flag for INT2.5
3	INTx4	R/W	0h	Interrupt Flag for INT2.4
2	INTx3	R/W	0h	Interrupt Flag for INT2.3
1	INTx2	R/W	0h	Interrupt Flag for INT2.2
0	INTx1	R/W	0h	Interrupt Flag for INT2.1

2.14.3.7 PIEIER3 Register (Offset = 6h) [reset = 0h]

PIEIER3 is shown in [Figure 2-34](#) and described in [Table 2-30](#).

INT3 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-34. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-30. PIEIER3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT3.16
14	INTx15	R/W	0h	Interrupt Enable for INT3.15
13	INTx14	R/W	0h	Interrupt Enable for INT3.14
12	INTx13	R/W	0h	Interrupt Enable for INT3.13
11	INTx12	R/W	0h	Interrupt Enable for INT3.12
10	INTx11	R/W	0h	Interrupt Enable for INT3.11
9	INTx10	R/W	0h	Interrupt Enable for INT3.10
8	INTx9	R/W	0h	Interrupt Enable for INT3.9
7	INTx8	R/W	0h	Interrupt Enable for INT3.8
6	INTx7	R/W	0h	Interrupt Enable for INT3.7
5	INTx6	R/W	0h	Interrupt Enable for INT3.6
4	INTx5	R/W	0h	Interrupt Enable for INT3.5
3	INTx4	R/W	0h	Interrupt Enable for INT3.4
2	INTx3	R/W	0h	Interrupt Enable for INT3.3
1	INTx2	R/W	0h	Interrupt Enable for INT3.2
0	INTx1	R/W	0h	Interrupt Enable for INT3.1

2.14.3.8 PIEIFR3 Register (Offset = 7h) [reset = 0h]

PIEIFR3 is shown in [Figure 2-35](#) and described in [Table 2-31](#).

INT3 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-35. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-31. PIEIFR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT3.16
14	INTx15	R/W	0h	Interrupt Flag for INT3.15
13	INTx14	R/W	0h	Interrupt Flag for INT3.14
12	INTx13	R/W	0h	Interrupt Flag for INT3.13
11	INTx12	R/W	0h	Interrupt Flag for INT3.12
10	INTx11	R/W	0h	Interrupt Flag for INT3.11
9	INTx10	R/W	0h	Interrupt Flag for INT3.10
8	INTx9	R/W	0h	Interrupt Flag for INT3.9
7	INTx8	R/W	0h	Interrupt Flag for INT3.8
6	INTx7	R/W	0h	Interrupt Flag for INT3.7
5	INTx6	R/W	0h	Interrupt Flag for INT3.6
4	INTx5	R/W	0h	Interrupt Flag for INT3.5
3	INTx4	R/W	0h	Interrupt Flag for INT3.4
2	INTx3	R/W	0h	Interrupt Flag for INT3.3
1	INTx2	R/W	0h	Interrupt Flag for INT3.2
0	INTx1	R/W	0h	Interrupt Flag for INT3.1

2.14.3.9 PIEIER4 Register (Offset = 8h) [reset = 0h]

PIEIER4 is shown in [Figure 2-36](#) and described in [Table 2-32](#).

INT4 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-36. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-32. PIEIER4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT4.16
14	INTx15	R/W	0h	Interrupt Enable for INT4.15
13	INTx14	R/W	0h	Interrupt Enable for INT4.14
12	INTx13	R/W	0h	Interrupt Enable for INT4.13
11	INTx12	R/W	0h	Interrupt Enable for INT4.12
10	INTx11	R/W	0h	Interrupt Enable for INT4.11
9	INTx10	R/W	0h	Interrupt Enable for INT4.10
8	INTx9	R/W	0h	Interrupt Enable for INT4.9
7	INTx8	R/W	0h	Interrupt Enable for INT4.8
6	INTx7	R/W	0h	Interrupt Enable for INT4.7
5	INTx6	R/W	0h	Interrupt Enable for INT4.6
4	INTx5	R/W	0h	Interrupt Enable for INT4.5
3	INTx4	R/W	0h	Interrupt Enable for INT4.4
2	INTx3	R/W	0h	Interrupt Enable for INT4.3
1	INTx2	R/W	0h	Interrupt Enable for INT4.2
0	INTx1	R/W	0h	Interrupt Enable for INT4.1

2.14.3.10 PIEIFR4 Register (Offset = 9h) [reset = 0h]

PIEIFR4 is shown in [Figure 2-37](#) and described in [Table 2-33](#).

INT4 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-37. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-33. PIEIFR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT4.16
14	INTx15	R/W	0h	Interrupt Flag for INT4.15
13	INTx14	R/W	0h	Interrupt Flag for INT4.14
12	INTx13	R/W	0h	Interrupt Flag for INT4.13
11	INTx12	R/W	0h	Interrupt Flag for INT4.12
10	INTx11	R/W	0h	Interrupt Flag for INT4.11
9	INTx10	R/W	0h	Interrupt Flag for INT4.10
8	INTx9	R/W	0h	Interrupt Flag for INT4.9
7	INTx8	R/W	0h	Interrupt Flag for INT4.8
6	INTx7	R/W	0h	Interrupt Flag for INT4.7
5	INTx6	R/W	0h	Interrupt Flag for INT4.6
4	INTx5	R/W	0h	Interrupt Flag for INT4.5
3	INTx4	R/W	0h	Interrupt Flag for INT4.4
2	INTx3	R/W	0h	Interrupt Flag for INT4.3
1	INTx2	R/W	0h	Interrupt Flag for INT4.2
0	INTx1	R/W	0h	Interrupt Flag for INT4.1

2.14.3.11 PIEIER5 Register (Offset = Ah) [reset = 0h]

PIEIER5 is shown in [Figure 2-38](#) and described in [Table 2-34](#).

INT5 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-38. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-34. PIEIER5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT5.16
14	INTx15	R/W	0h	Interrupt Enable for INT5.15
13	INTx14	R/W	0h	Interrupt Enable for INT5.14
12	INTx13	R/W	0h	Interrupt Enable for INT5.13
11	INTx12	R/W	0h	Interrupt Enable for INT5.12
10	INTx11	R/W	0h	Interrupt Enable for INT5.11
9	INTx10	R/W	0h	Interrupt Enable for INT5.10
8	INTx9	R/W	0h	Interrupt Enable for INT5.9
7	INTx8	R/W	0h	Interrupt Enable for INT5.8
6	INTx7	R/W	0h	Interrupt Enable for INT5.7
5	INTx6	R/W	0h	Interrupt Enable for INT5.6
4	INTx5	R/W	0h	Interrupt Enable for INT5.5
3	INTx4	R/W	0h	Interrupt Enable for INT5.4
2	INTx3	R/W	0h	Interrupt Enable for INT5.3
1	INTx2	R/W	0h	Interrupt Enable for INT5.2
0	INTx1	R/W	0h	Interrupt Enable for INT5.1

2.14.3.12 PIEIFR5 Register (Offset = Bh) [reset = 0h]

PIEIFR5 is shown in [Figure 2-39](#) and described in [Table 2-35](#).

INT5 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-39. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-35. PIEIFR5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT5.16
14	INTx15	R/W	0h	Interrupt Flag for INT5.15
13	INTx14	R/W	0h	Interrupt Flag for INT5.14
12	INTx13	R/W	0h	Interrupt Flag for INT5.13
11	INTx12	R/W	0h	Interrupt Flag for INT5.12
10	INTx11	R/W	0h	Interrupt Flag for INT5.11
9	INTx10	R/W	0h	Interrupt Flag for INT5.10
8	INTx9	R/W	0h	Interrupt Flag for INT5.9
7	INTx8	R/W	0h	Interrupt Flag for INT5.8
6	INTx7	R/W	0h	Interrupt Flag for INT5.7
5	INTx6	R/W	0h	Interrupt Flag for INT5.6
4	INTx5	R/W	0h	Interrupt Flag for INT5.5
3	INTx4	R/W	0h	Interrupt Flag for INT5.4
2	INTx3	R/W	0h	Interrupt Flag for INT5.3
1	INTx2	R/W	0h	Interrupt Flag for INT5.2
0	INTx1	R/W	0h	Interrupt Flag for INT5.1

2.14.3.13 PIEIER6 Register (Offset = Ch) [reset = 0h]

PIEIER6 is shown in [Figure 2-40](#) and described in [Table 2-36](#).

INT6 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-40. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-36. PIEIER6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT6.16
14	INTx15	R/W	0h	Interrupt Enable for INT6.15
13	INTx14	R/W	0h	Interrupt Enable for INT6.14
12	INTx13	R/W	0h	Interrupt Enable for INT6.13
11	INTx12	R/W	0h	Interrupt Enable for INT6.12
10	INTx11	R/W	0h	Interrupt Enable for INT6.11
9	INTx10	R/W	0h	Interrupt Enable for INT6.10
8	INTx9	R/W	0h	Interrupt Enable for INT6.9
7	INTx8	R/W	0h	Interrupt Enable for INT6.8
6	INTx7	R/W	0h	Interrupt Enable for INT6.7
5	INTx6	R/W	0h	Interrupt Enable for INT6.6
4	INTx5	R/W	0h	Interrupt Enable for INT6.5
3	INTx4	R/W	0h	Interrupt Enable for INT6.4
2	INTx3	R/W	0h	Interrupt Enable for INT6.3
1	INTx2	R/W	0h	Interrupt Enable for INT6.2
0	INTx1	R/W	0h	Interrupt Enable for INT6.1

2.14.3.14 PIEIFR6 Register (Offset = Dh) [reset = 0h]

PIEIFR6 is shown in [Figure 2-41](#) and described in [Table 2-37](#).

INT6 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-41. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-37. PIEIFR6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT6.16
14	INTx15	R/W	0h	Interrupt Flag for INT6.15
13	INTx14	R/W	0h	Interrupt Flag for INT6.14
12	INTx13	R/W	0h	Interrupt Flag for INT6.13
11	INTx12	R/W	0h	Interrupt Flag for INT6.12
10	INTx11	R/W	0h	Interrupt Flag for INT6.11
9	INTx10	R/W	0h	Interrupt Flag for INT6.10
8	INTx9	R/W	0h	Interrupt Flag for INT6.9
7	INTx8	R/W	0h	Interrupt Flag for INT6.8
6	INTx7	R/W	0h	Interrupt Flag for INT6.7
5	INTx6	R/W	0h	Interrupt Flag for INT6.6
4	INTx5	R/W	0h	Interrupt Flag for INT6.5
3	INTx4	R/W	0h	Interrupt Flag for INT6.4
2	INTx3	R/W	0h	Interrupt Flag for INT6.3
1	INTx2	R/W	0h	Interrupt Flag for INT6.2
0	INTx1	R/W	0h	Interrupt Flag for INT6.1

2.14.3.15 PIEIER7 Register (Offset = Eh) [reset = 0h]

PIEIER7 is shown in [Figure 2-42](#) and described in [Table 2-38](#).

INT7 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-42. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-38. PIEIER7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT7.16
14	INTx15	R/W	0h	Interrupt Enable for INT7.15
13	INTx14	R/W	0h	Interrupt Enable for INT7.14
12	INTx13	R/W	0h	Interrupt Enable for INT7.13
11	INTx12	R/W	0h	Interrupt Enable for INT7.12
10	INTx11	R/W	0h	Interrupt Enable for INT7.11
9	INTx10	R/W	0h	Interrupt Enable for INT7.10
8	INTx9	R/W	0h	Interrupt Enable for INT7.9
7	INTx8	R/W	0h	Interrupt Enable for INT7.8
6	INTx7	R/W	0h	Interrupt Enable for INT7.7
5	INTx6	R/W	0h	Interrupt Enable for INT7.6
4	INTx5	R/W	0h	Interrupt Enable for INT7.5
3	INTx4	R/W	0h	Interrupt Enable for INT7.4
2	INTx3	R/W	0h	Interrupt Enable for INT7.3
1	INTx2	R/W	0h	Interrupt Enable for INT7.2
0	INTx1	R/W	0h	Interrupt Enable for INT7.1

2.14.3.16 PIEIFR7 Register (Offset = Fh) [reset = 0h]

PIEIFR7 is shown in [Figure 2-43](#) and described in [Table 2-39](#).

INT7 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-43. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-39. PIEIFR7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT7.16
14	INTx15	R/W	0h	Interrupt Flag for INT7.15
13	INTx14	R/W	0h	Interrupt Flag for INT7.14
12	INTx13	R/W	0h	Interrupt Flag for INT7.13
11	INTx12	R/W	0h	Interrupt Flag for INT7.12
10	INTx11	R/W	0h	Interrupt Flag for INT7.11
9	INTx10	R/W	0h	Interrupt Flag for INT7.10
8	INTx9	R/W	0h	Interrupt Flag for INT7.9
7	INTx8	R/W	0h	Interrupt Flag for INT7.8
6	INTx7	R/W	0h	Interrupt Flag for INT7.7
5	INTx6	R/W	0h	Interrupt Flag for INT7.6
4	INTx5	R/W	0h	Interrupt Flag for INT7.5
3	INTx4	R/W	0h	Interrupt Flag for INT7.4
2	INTx3	R/W	0h	Interrupt Flag for INT7.3
1	INTx2	R/W	0h	Interrupt Flag for INT7.2
0	INTx1	R/W	0h	Interrupt Flag for INT7.1

2.14.3.17 PIEIER8 Register (Offset = 10h) [reset = 0h]

PIEIER8 is shown in [Figure 2-44](#) and described in [Table 2-40](#).

INT8 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-44. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-40. PIEIER8 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT8.16
14	INTx15	R/W	0h	Interrupt Enable for INT8.15
13	INTx14	R/W	0h	Interrupt Enable for INT8.14
12	INTx13	R/W	0h	Interrupt Enable for INT8.13
11	INTx12	R/W	0h	Interrupt Enable for INT8.12
10	INTx11	R/W	0h	Interrupt Enable for INT8.11
9	INTx10	R/W	0h	Interrupt Enable for INT8.10
8	INTx9	R/W	0h	Interrupt Enable for INT8.9
7	INTx8	R/W	0h	Interrupt Enable for INT8.8
6	INTx7	R/W	0h	Interrupt Enable for INT8.7
5	INTx6	R/W	0h	Interrupt Enable for INT8.6
4	INTx5	R/W	0h	Interrupt Enable for INT8.5
3	INTx4	R/W	0h	Interrupt Enable for INT8.4
2	INTx3	R/W	0h	Interrupt Enable for INT8.3
1	INTx2	R/W	0h	Interrupt Enable for INT8.2
0	INTx1	R/W	0h	Interrupt Enable for INT8.1

2.14.3.18 PIEIFR8 Register (Offset = 11h) [reset = 0h]

PIEIFR8 is shown in [Figure 2-45](#) and described in [Table 2-41](#).

INT8 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-45. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-41. PIEIFR8 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT8.16
14	INTx15	R/W	0h	Interrupt Flag for INT8.15
13	INTx14	R/W	0h	Interrupt Flag for INT8.14
12	INTx13	R/W	0h	Interrupt Flag for INT8.13
11	INTx12	R/W	0h	Interrupt Flag for INT8.12
10	INTx11	R/W	0h	Interrupt Flag for INT8.11
9	INTx10	R/W	0h	Interrupt Flag for INT8.10
8	INTx9	R/W	0h	Interrupt Flag for INT8.9
7	INTx8	R/W	0h	Interrupt Flag for INT8.8
6	INTx7	R/W	0h	Interrupt Flag for INT8.7
5	INTx6	R/W	0h	Interrupt Flag for INT8.6
4	INTx5	R/W	0h	Interrupt Flag for INT8.5
3	INTx4	R/W	0h	Interrupt Flag for INT8.4
2	INTx3	R/W	0h	Interrupt Flag for INT8.3
1	INTx2	R/W	0h	Interrupt Flag for INT8.2
0	INTx1	R/W	0h	Interrupt Flag for INT8.1

2.14.3.19 PIEIER9 Register (Offset = 12h) [reset = 0h]

PIEIER9 is shown in [Figure 2-46](#) and described in [Table 2-42](#).

INT9 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-46. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-42. PIEIER9 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT9.16
14	INTx15	R/W	0h	Interrupt Enable for INT9.15
13	INTx14	R/W	0h	Interrupt Enable for INT9.14
12	INTx13	R/W	0h	Interrupt Enable for INT9.13
11	INTx12	R/W	0h	Interrupt Enable for INT9.12
10	INTx11	R/W	0h	Interrupt Enable for INT9.11
9	INTx10	R/W	0h	Interrupt Enable for INT9.10
8	INTx9	R/W	0h	Interrupt Enable for INT9.9
7	INTx8	R/W	0h	Interrupt Enable for INT9.8
6	INTx7	R/W	0h	Interrupt Enable for INT9.7
5	INTx6	R/W	0h	Interrupt Enable for INT9.6
4	INTx5	R/W	0h	Interrupt Enable for INT9.5
3	INTx4	R/W	0h	Interrupt Enable for INT9.4
2	INTx3	R/W	0h	Interrupt Enable for INT9.3
1	INTx2	R/W	0h	Interrupt Enable for INT9.2
0	INTx1	R/W	0h	Interrupt Enable for INT9.1

2.14.3.20 PIEIFR9 Register (Offset = 13h) [reset = 0h]

PIEIFR9 is shown in [Figure 2-47](#) and described in [Table 2-43](#).

INT9 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-47. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-43. PIEIFR9 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT9.16
14	INTx15	R/W	0h	Interrupt Flag for INT9.15
13	INTx14	R/W	0h	Interrupt Flag for INT9.14
12	INTx13	R/W	0h	Interrupt Flag for INT9.13
11	INTx12	R/W	0h	Interrupt Flag for INT9.12
10	INTx11	R/W	0h	Interrupt Flag for INT9.11
9	INTx10	R/W	0h	Interrupt Flag for INT9.10
8	INTx9	R/W	0h	Interrupt Flag for INT9.9
7	INTx8	R/W	0h	Interrupt Flag for INT9.8
6	INTx7	R/W	0h	Interrupt Flag for INT9.7
5	INTx6	R/W	0h	Interrupt Flag for INT9.6
4	INTx5	R/W	0h	Interrupt Flag for INT9.5
3	INTx4	R/W	0h	Interrupt Flag for INT9.4
2	INTx3	R/W	0h	Interrupt Flag for INT9.3
1	INTx2	R/W	0h	Interrupt Flag for INT9.2
0	INTx1	R/W	0h	Interrupt Flag for INT9.1

2.14.3.21 PIEIER10 Register (Offset = 14h) [reset = 0h]

PIEIER10 is shown in [Figure 2-48](#) and described in [Table 2-44](#).

INT10 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-48. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-44. PIEIER10 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT10.16
14	INTx15	R/W	0h	Interrupt Enable for INT10.15
13	INTx14	R/W	0h	Interrupt Enable for INT10.14
12	INTx13	R/W	0h	Interrupt Enable for INT10.13
11	INTx12	R/W	0h	Interrupt Enable for INT10.12
10	INTx11	R/W	0h	Interrupt Enable for INT10.11
9	INTx10	R/W	0h	Interrupt Enable for INT10.10
8	INTx9	R/W	0h	Interrupt Enable for INT10.9
7	INTx8	R/W	0h	Interrupt Enable for INT10.8
6	INTx7	R/W	0h	Interrupt Enable for INT10.7
5	INTx6	R/W	0h	Interrupt Enable for INT10.6
4	INTx5	R/W	0h	Interrupt Enable for INT10.5
3	INTx4	R/W	0h	Interrupt Enable for INT10.4
2	INTx3	R/W	0h	Interrupt Enable for INT10.3
1	INTx2	R/W	0h	Interrupt Enable for INT10.2
0	INTx1	R/W	0h	Interrupt Enable for INT10.1

2.14.3.22 PIEIFR10 Register (Offset = 15h) [reset = 0h]

PIEIFR10 is shown in [Figure 2-49](#) and described in [Table 2-45](#).

INT10 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-49. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-45. PIEIFR10 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT10.16
14	INTx15	R/W	0h	Interrupt Flag for INT10.15
13	INTx14	R/W	0h	Interrupt Flag for INT10.14
12	INTx13	R/W	0h	Interrupt Flag for INT10.13
11	INTx12	R/W	0h	Interrupt Flag for INT10.12
10	INTx11	R/W	0h	Interrupt Flag for INT10.11
9	INTx10	R/W	0h	Interrupt Flag for INT10.10
8	INTx9	R/W	0h	Interrupt Flag for INT10.9
7	INTx8	R/W	0h	Interrupt Flag for INT10.8
6	INTx7	R/W	0h	Interrupt Flag for INT10.7
5	INTx6	R/W	0h	Interrupt Flag for INT10.6
4	INTx5	R/W	0h	Interrupt Flag for INT10.5
3	INTx4	R/W	0h	Interrupt Flag for INT10.4
2	INTx3	R/W	0h	Interrupt Flag for INT10.3
1	INTx2	R/W	0h	Interrupt Flag for INT10.2
0	INTx1	R/W	0h	Interrupt Flag for INT10.1

2.14.3.23 PIEIER11 Register (Offset = 16h) [reset = 0h]

PIEIER11 is shown in [Figure 2-50](#) and described in [Table 2-46](#).

INT11 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-50. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-46. PIEIER11 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT11.16
14	INTx15	R/W	0h	Interrupt Enable for INT11.15
13	INTx14	R/W	0h	Interrupt Enable for INT11.14
12	INTx13	R/W	0h	Interrupt Enable for INT11.13
11	INTx12	R/W	0h	Interrupt Enable for INT11.12
10	INTx11	R/W	0h	Interrupt Enable for INT11.11
9	INTx10	R/W	0h	Interrupt Enable for INT11.10
8	INTx9	R/W	0h	Interrupt Enable for INT11.9
7	INTx8	R/W	0h	Interrupt Enable for INT11.8
6	INTx7	R/W	0h	Interrupt Enable for INT11.7
5	INTx6	R/W	0h	Interrupt Enable for INT11.6
4	INTx5	R/W	0h	Interrupt Enable for INT11.5
3	INTx4	R/W	0h	Interrupt Enable for INT11.4
2	INTx3	R/W	0h	Interrupt Enable for INT11.3
1	INTx2	R/W	0h	Interrupt Enable for INT11.2
0	INTx1	R/W	0h	Interrupt Enable for INT11.1

2.14.3.24 PIEIFR11 Register (Offset = 17h) [reset = 0h]

PIEIFR11 is shown in [Figure 2-51](#) and described in [Table 2-47](#).

INT11 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-51. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-47. PIEIFR11 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT11.16
14	INTx15	R/W	0h	Interrupt Flag for INT11.15
13	INTx14	R/W	0h	Interrupt Flag for INT11.14
12	INTx13	R/W	0h	Interrupt Flag for INT11.13
11	INTx12	R/W	0h	Interrupt Flag for INT11.12
10	INTx11	R/W	0h	Interrupt Flag for INT11.11
9	INTx10	R/W	0h	Interrupt Flag for INT11.10
8	INTx9	R/W	0h	Interrupt Flag for INT11.9
7	INTx8	R/W	0h	Interrupt Flag for INT11.8
6	INTx7	R/W	0h	Interrupt Flag for INT11.7
5	INTx6	R/W	0h	Interrupt Flag for INT11.6
4	INTx5	R/W	0h	Interrupt Flag for INT11.5
3	INTx4	R/W	0h	Interrupt Flag for INT11.4
2	INTx3	R/W	0h	Interrupt Flag for INT11.3
1	INTx2	R/W	0h	Interrupt Flag for INT11.2
0	INTx1	R/W	0h	Interrupt Flag for INT11.1

2.14.3.25 PIEIER12 Register (Offset = 18h) [reset = 0h]

PIEIER12 is shown in [Figure 2-52](#) and described in [Table 2-48](#).

INT12 Group Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the core interrupt enable register.

Setting a bit to 1 will enable the servicing of the respective interrupt.

Setting a bit to 0 will disable the servicing of the bit.

Figure 2-52. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-48. PIEIER12 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Enable for INT12.16
14	INTx15	R/W	0h	Interrupt Enable for INT12.15
13	INTx14	R/W	0h	Interrupt Enable for INT12.14
12	INTx13	R/W	0h	Interrupt Enable for INT12.13
11	INTx12	R/W	0h	Interrupt Enable for INT12.12
10	INTx11	R/W	0h	Interrupt Enable for INT12.11
9	INTx10	R/W	0h	Interrupt Enable for INT12.10
8	INTx9	R/W	0h	Interrupt Enable for INT12.9
7	INTx8	R/W	0h	Interrupt Enable for INT12.8
6	INTx7	R/W	0h	Interrupt Enable for INT12.7
5	INTx6	R/W	0h	Interrupt Enable for INT12.6
4	INTx5	R/W	0h	Interrupt Enable for INT12.5
3	INTx4	R/W	0h	Interrupt Enable for INT12.4
2	INTx3	R/W	0h	Interrupt Enable for INT12.3
1	INTx2	R/W	0h	Interrupt Enable for INT12.2
0	INTx1	R/W	0h	Interrupt Enable for INT12.1

2.14.3.26 PIEIFR12 Register (Offset = 19h) [reset = 0h]

PIEIFR12 is shown in [Figure 2-53](#) and described in [Table 2-49](#).

INT12 Group Flag Register

These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register.

When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit.

This register can also be read to determine which interrupts are active or pending.

Figure 2-53. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-49. PIEIFR12 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	INTx16	R/W	0h	Interrupt Flag for INT12.16
14	INTx15	R/W	0h	Interrupt Flag for INT12.15
13	INTx14	R/W	0h	Interrupt Flag for INT12.14
12	INTx13	R/W	0h	Interrupt Flag for INT12.13
11	INTx12	R/W	0h	Interrupt Flag for INT12.12
10	INTx11	R/W	0h	Interrupt Flag for INT12.11
9	INTx10	R/W	0h	Interrupt Flag for INT12.10
8	INTx9	R/W	0h	Interrupt Flag for INT12.9
7	INTx8	R/W	0h	Interrupt Flag for INT12.8
6	INTx7	R/W	0h	Interrupt Flag for INT12.7
5	INTx6	R/W	0h	Interrupt Flag for INT12.6
4	INTx5	R/W	0h	Interrupt Flag for INT12.5
3	INTx4	R/W	0h	Interrupt Flag for INT12.4
2	INTx3	R/W	0h	Interrupt Flag for INT12.3
1	INTx2	R/W	0h	Interrupt Flag for INT12.2
0	INTx1	R/W	0h	Interrupt Flag for INT12.1

2.14.4 WD_REGS Registers

[Table 2-50](#) lists the memory-mapped registers for the WD_REGS. All register offset addresses not listed in [Table 2-50](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-50. WD_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
22h	SCSR	System Control & Status Register	EALLOW	Go
23h	WDCNTR	Watchdog Counter Register	EALLOW	Go
25h	WDKEY	Watchdog Reset Key Register	EALLOW	Go
29h	WDCR	Watchdog Control Register	EALLOW	Go
2Ah	WDWCR	Watchdog Windowed Control Register	EALLOW	Go

2.14.4.1 SCSR Register (Offset = 22h) [reset = 5h]

SCSR is shown in [Figure 2-54](#) and described in [Table 2-51](#).

System Control & Status Register

Figure 2-54. 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/W=1-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-51. SCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R=0	0h	Reserved
2	WDINTS	R	1h	Watchdog interrupt (WDINTn) status signal. This is a read only bit reflecting the current state of the WDINTn signal from the watchdog block (after synchronization with SYSCLKOUT). If this bit is 1, the watchdog interrupt is not active. If this bit is 0, then the watchdog interrupt is active. Note: ,If the WDINTn signal is used to wake up from IDLE or STANDBY condition, then the user should make sure that the WDINTn signal goes back high again before attempting to go back into IDLE or STANDBY mode. Reading this bit will tell the user the current state of this signal.
1	WDENINT	R/W	0h	If this bit is set to 1, the watchdog reset (WDRSTn) output signal is disabled and the watchdog interrupt (WDINTn) output signal is enabled. If this bit is zero, then the WDRSTn output signal is enabled and the WDINTn output signal is disabled. This is the default state on system reset (SYSRSn).
0	WDOVERRIDE	R/W=1	1h	If this bit is set to 1, the user is allowed to change the state of the Watchdog disable (WDDIS) bit in the Watchdog Control (WDCR) register. If the WDOVERRIDE bit is cleared, by writing a 1 the WDDIS bit cannot be modified by the user. Writing a 0 will have no effect. If this bit is cleared, then it will remain in this state until a reset occurs. The current state of this bit is readable by the user.

2.14.4.2 WDCNTR Register (Offset = 23h) [reset = 0h]

WDCNTR is shown in [Figure 2-55](#) and described in [Table 2-52](#).

Watchdog Counter Register

Figure 2-55. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-52. WDCNTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-0	WDCNTR	R	0h	These bits contain the current value of the WD counter. The 8-bit counter continually increments at the WDCLK rate. If the counter overflows, then a watchdog output pulse (WDOUTn) is generated. If the WDKEY register is written with a valid combination, then the counter is reset to zero.

2.14.4.3 WDKEY Register (Offset = 25h) [reset = 0h]

WDKEY is shown in [Figure 2-56](#) and described in [Table 2-53](#).

Watchdog Reset Key Register

Figure 2-56. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-53. WDKEY Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-0	WDKEY	R/W	0h	Writing 0x55 followed by 0xAA will cause the WDCNTR bits to be cleared. Note: [1] Reads from the WDKEY return the value of WDCR register.

2.14.4.4 WDCR Register (Offset = 29h) [reset = 0h]

WDCR is shown in [Figure 2-57](#) and described in [Table 2-54](#).

Watchdog Control Register

Figure 2-57. WDCR Register

15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED	WDDIS	WDCHK			WDPS		
R-0h	R/W-0h	R=0/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-54. WDCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	WDDIS	R/W	0h	Writing a 1 to this bit will disable the watchdog module. Writing a 0 will enable the module. This bit can only be modified if the WDOVERRIDE bit in the SCSR2 register is set to 1. On reset, the watchdog module is enabled.
5-3	WDCHK	R=0/W	0h	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).
2-0	WDPS	R/W	0h	These bits configure the watchdog counter clock (WDCLK) rate relative to INTOSC1/512: 000 WDCLK = INTOSC1/512/1 001 WDCLK = INTOSC1/512/1 010 WDCLK = INTOSC1/512/2 011 WDCLK = INTOSC1/512/4 100 WDCLK = INTOSC1/512/8 101 WDCLK = INTOSC1/512/16 110 WDCLK = INTOSC1/512/32 111 WDCLK = INTOSC1/512/64

2.14.4.5 WDWCRC Register (Offset = 2Ah) [reset = 0h]

WDWCRC is shown in [Figure 2-58](#) and described in [Table 2-55](#).

Watchdog Windowed Control Register

Figure 2-58. WDWCRC Register

15	14	13	12	11	10	9	8
RESERVED							FIRSTKEY
R=0-0h							R-0h
7	6	5	4	3	2	1	0
MIN							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-55. WDWCRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	RESERVED	R=0	0h	Reserved
8	FIRSTKEY	R	0h	<p>This bit indicates if the 1st valid WDKEY (0x55 + 0xAA) got detected after MIN was configured to a non-zero value</p> <p>0: First Valid Key after non-zero MIN configuration has not happened yet</p> <p>1: First Valid key after non-zero MIN configuration got detected</p> <p>Notes:</p> <p>[1] If MIN = 0, this bit is never set</p> <p>[2] If MIN is changed back to 0x0 from a non-zero value, this bit is auto-cleared</p> <p>[3] This bit is added for debug purposes only</p>
7-0	MIN	R/W	0h	These bits define the lower limit of the Windowed functionality

2.14.5 NMI_INTRUPT_REGS Registers

Table 2-56 lists the memory-mapped registers for the NMI_INTRUPT_REGS. All register offset addresses not listed in Table 2-56 should be considered as reserved locations and the register contents should not be modified.

Table 2-56. NMI_INTRUPT_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	NMICFG	NMI Configuration Register	EALLOW	Go
1h	NMIFLG	NMI Flag Register (XRSn Clear)		Go
2h	NMIFLGCLR	NMI Flag Clear Register	EALLOW	Go
3h	NMIFLGFRG	NMI Flag Force Register	EALLOW	Go
4h	NMIWDCNT	NMI Watchdog Counter Register		Go
5h	NMIWDPRD	NMI Watchdog Period Register	EALLOW	Go
6h	NMISHDFLG	NMI Shadow Flag Register		Go

2.14.5.1 NMICFG Register (Offset = 0h) [reset = 0h]

NMICFG is shown in [Figure 2-59](#) and described in [Table 2-57](#).

NMI Configuration Register

Figure 2-59. 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/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-57. NMICFG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R=0	0h	Reserved
0	NMIE	R/W=1	0h	When set to 1 any condition will generate an NMI interrupt to the C28 CPU and kick off the NMI watchdog counter. As part of boot sequence this bit should be set after the device security related initialization is complete. 0 NMI disabled 1 NMI enabled

2.14.5.2 NMIFLG Register (Offset = 1h) [reset = 0h]

NMIFLG is shown in [Figure 2-60](#) and described in [Table 2-58](#).

NMI Flag Register (XRSn Clear)

Figure 2-60. NMIFLG Register

15	14	13	12	11	10	9	8
RESERVED				RESERVED	CPU2NMIWDRSn	CPU2WDRSn	RESERVED
R=0-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	CPU2HWBISTERR	CPU1HWBISTERR	FLUNCERR	RAMUNCERR	CLOCKFAIL	NMIINT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-58. NMIFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R=0	0h	Reserved
11	RESERVED	R	0h	Reserved
10	CPU2NMIWDRSn	R	0h	CPU2 NMIWDRSn Reset Indication Flag: This bits indicates if CPU2s NMIWDRSn was fired or not. 0 No CPU2.NMIWDRsn was fired 1 CPU2.NMIWDRSn was fired to CPU2 Note: [1] This bits is reserved for CPU2.NMIFLG register
9	CPU2WDRSn	R	0h	CPU2 WDRSn Reset Indication Flag: This bits indicates if CPU2s WDRSn was fired or not. 0 No CPU2.WDRsn was fired 1 CPU2.WDRSn was fired to CPU2 Note: [1] This bits is reserved for CPU2.NMIFLG register
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R	0h	PIE Vector Fetch Error Flag: This bit indicates if an error occurred on an Vector Fect by the other CPU in the device. For example, CPU1.NMIWD gets an NMI on an Vector fetch Error on CPU2. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset: 0,No Vector Fetch Error condition (on the other CPU) pending 1,Vector Fetch error condition (on the other CPU) generated
5	CPU2HWBISTERR	R	0h	HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of C28 CPU2 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset: 0,No C28 HWBIST error condition pending 1,C28 BIST error condition generated
4	CPU1HWBISTERR	R	0h	HW BIST Error NMI Flag: This bit indicates if the time out error or a signature mismatch error condition during hardware BIST of C28 CPU1 occurred. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset: 0,No C28 HWBIST error condition pending 1,C28 BIST error condition generated

Table 2-58. NMIFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	FLUNCERR	R	0h	Flash Uncorrectable Error NMI Flag: This bit indicates if an uncorrectable error occurred on a C28 Flash access and that condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset: 0, No C28 Flash uncorrectable error condition pending 1, C28 Flash uncorrectable error condition generated
2	RAMUNCERR	R	0h	RAM Uncorrectable Error NMI Flag: This bit indicates if an uncorrectable error occurred on a RAM access (by any master) and that condition is latched. This bit can only be cleared by the user writing to the corresponding clear bit in the NMIFLGCLR register or by an XRSn reset: 0, No RAM uncorrectable error condition pending 1, RAM uncorrectable error condition generated
1	CLOCKFAIL	R	0h	Clock Fail Interrupt Flag: These bits indicate if the CLOCKFAIL condition is latched. These bits can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an XRSn reset: 0, No CLOCKFAIL Condition Pending 1, CLOCKFAIL Condition Generated
0	NMIINT	R	0h	NMI Interrupt Flag: This bit indicates if an NMI interrupt was generated. This bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an XRSn reset: 0 No NMI Interrupt Generated 1 NMI Interrupt Generated No further NMI interrupts pulses are generated until this flag is cleared by the user.

2.14.5.3 NMIFLGCLR Register (Offset = 2h) [reset = 0h]

NMIFLGCLR is shown in [Figure 2-61](#) and described in [Table 2-59](#).

NMI Flag Clear Register

Figure 2-61. NMIFLGCLR Register

15	14	13	12	11	10	9	8
RESERVED				OVF	CPU2NMIWDR Sn	CPU2WDRSn	RESERVED
R=0-0h				R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	CPU2HWBIST ERR	CPU1HWBIST ERR	FLUNCERR	RAMUNCERR	CLOCKFAIL	NMIINT
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-59. NMIFLGCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R=0	0h	Reserved
11	OVF	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
10	CPU2NMIWDRSn	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
9	CPU2WDRSn	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag. [3] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.

Table 2-59. NMIFLGCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	CPU2HWBISTERR	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
4	CPU1HWBISTERR	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
3	FLUNCERR	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
2	RAMUNCERR	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
1	CLOCKFAIL	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.
0	NMIINT	R=0/W=1	0h	Writing a 1 to the respective bit clears the corresponding flag bit in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. Notes: [1] If hardware is trying to set a bit to 1 while software is trying to clear a bit to 0 on the same cycle, hardware has priority. [2] Users should clear the pending FAIL flag first and then clear the NMIINT flag.

2.14.5.4 NMIFLGFR Register (Offset = 3h) [reset = 0h]

NMIFLGFR is shown in [Figure 2-62](#) and described in [Table 2-60](#).

NMI Flag Force Register

Figure 2-62. NMIFLGFR Register

15	14	13	12	11	10	9	8
RESERVED				OVF	CPU2NMIWDR Sn	CPU2WDRSn	RESERVED
R=0-0h				R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	CPU2HWBIST ERR	CPU1HWBIST ERR	FLUNCERR	RAMUNCERR	CLOCKFAIL	RESERVED
R=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=0-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-60. NMIFLGFR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R=0	0h	Reserved
11	OVF	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
10	CPU2NMIWDRSn	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms. Note: [1] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers
9	CPU2WDRSn	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms. Note: [1] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
5	CPU2HWBISTERR	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
4	CPU1HWBISTERR	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
3	FLUNCERR	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.

Table 2-60. NMIFLGFRG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	RAMUNCERR	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
1	CLOCKFAIL	R=0/W=1	0h	Writing a 1 to these bits will set the respective FAIL flag in the NMIFLG and NMISHDFLG registers. Writes of 0 are ignored. Always reads back 0. This can be used as a means to test the NMI mechanisms.
0	RESERVED	R=0	0h	Reserved

2.14.5.5 NMIWDCNT Register (Offset = 4h) [reset = 0h]

NMIWDCNT is shown in [Figure 2-63](#) and described in [Table 2-61](#).

NMI Watchdog Counter Register

Figure 2-63. NMIWDCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMIWDCNT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-61. NMIWDCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NMIWDCNT	R	0h	<p>NMI Watchdog Counter: This 16-bit incremental counter will start incrementing whenever any one of the enabled FAIL flags are set. If the counter reaches the period value, an NMIRSn signal is fired which will then resets the system. The counter will reset to zero when it reaches the period value and will then restart counting if any of the enabled FAIL flags are set.</p> <p>If no enabled FAIL flag is set, then the counter will reset to zero and remain at zero until an enabled FAIL flag is set.</p> <p>Normally, the software would respond to the NMI interrupt generated and clear the offending FLAG(s) before the NMI watchdog triggers a reset. In some situations, the software may decide to allow the watchdog to reset the device anyway.</p> <p>The counter is clocked at the SYSCLKOUT rate.</p>

2.14.5.6 NMIWDPRD Register (Offset = 5h) [reset = FFFFh]

NMIWDPRD is shown in [Figure 2-64](#) and described in [Table 2-62](#).

NMI Watchdog Period Register

Figure 2-64. NMIWDPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMIWDPRD															
R/W-FFFFh															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-62. NMIWDPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	NMIWDPRD	R/W	FFFFh	<p>NMI Watchdog Period: This 16-bit value contains the period value at which a reset is generated when the watchdog counter matches. At reset this value is set at the maximum. The software can decrease the period value at initialization time.</p> <p>Writing a PERIOD value that is smaller then the current counter value will automatically force an NMIRSn and hence reset the watchdog counter.</p>

2.14.5.7 NMISHDFLG Register (Offset = 6h) [reset = 0h]

NMISHDFLG is shown in [Figure 2-65](#) and described in [Table 2-63](#).

NMI Shadow Flag Register

Figure 2-65. NMISHDFLG Register

15	14	13	12	11	10	9	8
RESERVED				OVF	CPU2NMIWDR Sn	CPU2WDRSn	RESERVED
R=0-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	PIEVECTERR	CPU2HWBIST ERR	CPU1HWBIST ERR	FLUNCERR	RAMUNCERR	CLOCKFAIL	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R=0-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-63. NMISHDFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R=0	0h	Reserved
11	OVF	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRC and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
10	CPU2NMIWDRSn	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRC and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
9	CPU2WDRSn	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRC and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean. [2] CPU2WDRSn and CPU2NMIWDRSn bits are reserved for CPU2.NMIFLGCLR registers
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	PIEVECTERR	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRC and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.

Table 2-63. NMISHDFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	CPU2HWBISTERR	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRM and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
4	CPU1HWBISTERR	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRM and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
3	FLUNCERR	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRM and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
2	RAMUNCERR	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRM and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
1	CLOCKFAIL	R	0h	Shadow NMI Flags: When an NMIFLG bit is set due to any of the possible NMI source in the device, the corresponding bit in this register is also set. Note that NMIFLGFRM and NMIFLGCLR register also affects the bits of this register in the same way as they do for the NMIFLG register. This register is resetted only by PORn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean.
0	RESERVED	R=0	0h	Reserved

2.14.6 XINT_REGS Registers

Table 2-64 lists the memory-mapped registers for the XINT_REGS. All register offset addresses not listed in Table 2-64 should be considered as reserved locations and the register contents should not be modified.

Table 2-64. XINT_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	XINT1CR	XINT1 configuration register		Go
1h	XINT2CR	XINT2 configuration register		Go
2h	XINT3CR	XINT3 configuration register		Go
3h	XINT4CR	XINT4 configuration register		Go
4h	XINT5CR	XINT5 configuration register		Go
8h	XINT1CTR	XINT1 counter register		Go
9h	XINT2CTR	XINT2 counter register		Go
Ah	XINT3CTR	XINT3 counter register		Go

2.14.6.1 XINT1CR Register (Offset = 0h) [reset = 0h]

XINT1CR is shown in [Figure 2-66](#) and described in [Table 2-65](#).

XINT1 configuration register

Figure 2-66. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-65. 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
1	RESERVED	R=0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled

2.14.6.2 XINT2CR Register (Offset = 1h) [reset = 0h]

XINT2CR is shown in [Figure 2-67](#) and described in [Table 2-66](#).

XINT2 configuration register

Figure 2-67. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-66. 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
1	RESERVED	R=0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled

2.14.6.3 XINT3CR Register (Offset = 2h) [reset = 0h]

XINT3CR is shown in [Figure 2-68](#) and described in [Table 2-67](#).

XINT3 configuration register

Figure 2-68. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-67. 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
1	RESERVED	R=0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled

2.14.6.4 XINT4CR Register (Offset = 3h) [reset = 0h]

XINT4CR is shown in [Figure 2-69](#) and described in [Table 2-68](#).

XINT4 configuration register

Figure 2-69. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-68. 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
1	RESERVED	R=0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled

2.14.6.5 XINT5CR Register (Offset = 4h) [reset = 0h]

XINT5CR is shown in [Figure 2-70](#) and described in [Table 2-69](#).

XINT5 configuration register

Figure 2-70. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-69. 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
1	RESERVED	R=0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled

2.14.6.6 XINT1CTR Register (Offset = 8h) [reset = 0h]

XINT1CTR is shown in [Figure 2-71](#) and described in [Table 2-70](#).

XINT1 counter register

Figure 2-71. XINT1CTR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCTR															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-70. 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.

2.14.6.7 XINT2CTR Register (Offset = 9h) [reset = 0h]

XINT2CTR is shown in [Figure 2-72](#) and described in [Table 2-71](#).

XINT2 counter register

Figure 2-72. XINT2CTR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCTR															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-71. XINT2CTR 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.

2.14.6.8 XINT3CTR Register (Offset = Ah) [reset = 0h]

XINT3CTR is shown in [Figure 2-73](#) and described in [Table 2-72](#).

XINT3 counter register

Figure 2-73. XINT3CTR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCTR															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-72. 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.

2.14.7 DMA_CLA_SRC_SEL_REGS Registers

Table 2-73 lists the memory-mapped registers for the DMA_CLA_SRC_SEL_REGS. All register offset addresses not listed in Table 2-73 should be considered as reserved locations and the register contents should not be modified.

Table 2-73. DMA_CLA_SRC_SEL_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	CLA1TASKSRCSELLOCK	CLA1 Task Trigger Source Select Lock Register	EALLOW	Go
4h	DMACHSRCSELLOCK	DMA Channel Triger Source Select Lock Register	EALLOW	Go
6h	CLA1TASKSRCSEL1	CLA1 Task Trigger Source Select Register-1	EALLOW	Go
8h	CLA1TASKSRCSEL2	CLA1 Task Trigger Source Select Register-2	EALLOW	Go
16h	DMACHSRCSEL1	DMA Channel Trigger Source Select Register-1	EALLOW	Go
18h	DMACHSRCSEL2	DMA Channel Trigger Source Select Register-2	EALLOW	Go

2.14.7.1 CLA1TASKSRCSELLOCK Register (Offset = 0h) [reset = 0h]

CLA1TASKSRCSELLOCK is shown in [Figure 2-74](#) and described in [Table 2-74](#).

CLA1 Task Trigger Source Select Lock Register

Figure 2-74. 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/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-74. CLA1TASKSRCSELLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	CLA1TASKSRCSEL2	R/SOnce	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
0	CLA1TASKSRCSEL1	R/SOnce	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

2.14.7.2 DMACHSRCSELLOCK Register (Offset = 4h) [reset = 0h]

DMACHSRCSELLOCK is shown in [Figure 2-75](#) and described in [Table 2-75](#).

DMA Channel Triger Source Select Lock Register

Figure 2-75. 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/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-75. DMACHSRCSELLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	DMACHSRCSEL2	R/SOnce	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
0	DMACHSRCSEL1	R/SOnce	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

2.14.7.3 CLA1TASKSRCSEL1 Register (Offset = 6h) [reset = 0h]

CLA1TASKSRCSEL1 is shown in [Figure 2-76](#) and described in [Table 2-76](#).

CLA1 Task Trigger Source Select Register-1

Figure 2-76. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-76. CLA1TASKSRCSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	TASK4	R/W	0h	Selects the Trigger Source for TASK4 of CLA1
23-16	TASK3	R/W	0h	Selects the Trigger Source for TASK3 of CLA1
15-8	TASK2	R/W	0h	Selects the Trigger Source for TASK2 of CLA1
7-0	TASK1	R/W	0h	Selects the Trigger Source for TASK1 of CLA1

2.14.7.4 CLA1TASKSRCSEL2 Register (Offset = 8h) [reset = 0h]

CLA1TASKSRCSEL2 is shown in [Figure 2-77](#) and described in [Table 2-77](#).

CLA1 Task Trigger Source Select Register-2

Figure 2-77. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-77. CLA1TASKSRCSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	TASK8	R/W	0h	Selects the Trigger Source for TASK8 of CLA1
23-16	TASK7	R/W	0h	Selects the Trigger Source for TASK7 of CLA1
15-8	TASK6	R/W	0h	Selects the Trigger Source for TASK6 of CLA1
7-0	TASK5	R/W	0h	Selects the Trigger Source for TASK5 of CLA1

2.14.7.5 DMACHSRCSEL1 Register (Offset = 16h) [reset = 0h]

DMACHSRCSEL1 is shown in [Figure 2-78](#) and described in [Table 2-78](#).

DMA Channel Trigger Source Select Register-1

Figure 2-78. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-78. DMACHSRCSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	CH4	R/W	0h	Selects the Trigger and Sync Source CH4 of DMA
23-16	CH3	R/W	0h	Selects the Trigger and Sync Source CH3 of DMA
15-8	CH2	R/W	0h	Selects the Trigger and Sync Source CH2 of DMA
7-0	CH1	R/W	0h	Selects the Trigger and Sync Source CH1 of DMA

2.14.7.6 DMACHSRCSEL2 Register (Offset = 18h) [reset = 0h]

DMACHSRCSEL2 is shown in [Figure 2-79](#) and described in [Table 2-79](#).

DMA Channel Trigger Source Select Register-2

Figure 2-79. 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									

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-79. 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
7-0	CH5	R/W	0h	Selects the Trigger and Sync Source CH5 of DMA

2.14.8 DEV_CFG_REGS Registers

Table 2-80 lists the memory-mapped registers for the DEV_CFG_REGS. All register offset addresses not listed in Table 2-80 should be considered as reserved locations and the register contents should not be modified.

Table 2-80. DEV_CFG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	DEVCFGLOCK1	Lock bit for CPUSELx registers	EALLOW	Go
8h	PARTIDL	Lower 32-bit of Device PART Identification Number		Go
Ah	PARTIDH	Upper 32-bit of Device PART Identification Number		Go
Ch	REVID	Device Revision Number		Go
10h	DC0	Device Capability: Device Information		Go
12h	DC1	Device Capability: Processing Block Customization		Go
14h	DC2	Device Capability: EMIF Customization		Go
16h	DC3	Device Capability: Peripheral Customization		Go
18h	DC4	Device Capability: Peripheral Customization		Go
1Ah	DC5	Device Capability: Peripheral Customization		Go
1Ch	DC6	Device Capability: Peripheral Customization		Go
1Eh	DC7	Device Capability: Peripheral Customization		Go
20h	DC8	Device Capability: Peripheral Customization		Go
22h	DC9	Device Capability: Peripheral Customization		Go
24h	DC10	Device Capability: Peripheral Customization		Go
26h	DC11	Device Capability: Peripheral Customization		Go
28h	DC12	Device Capability: Peripheral Customization		Go
2Ah	DC13	Device Capability: Peripheral Customization		Go
2Ch	DC14	Device Capability: Analog Modules Customization		Go
2Eh	DC15	Device Capability: Analog Modules Customization		Go
32h	DC17	Device Capability: Analog Modules Customization		Go
34h	DC18	Device Capability: CPU1 Lx SRAM Customization		Go
36h	DC19	Device Capability: CPU2 Lx SRAM Customization		Go
38h	DC20	Device Capability: GSx SRAM Customization		Go
60h	PERCNF1	Peripheral Configuration register		Go
74h	FUSEERR	e-Fuse error Status register		Go
82h	SOFTPRES0	Processing Block Software Reset register	EALLOW	Go
84h	SOFTPRES1	EMIF Software Reset register	EALLOW	Go
86h	SOFTPRES2	Peripheral Software Reset register	EALLOW	Go
88h	SOFTPRES3	Peripheral Software Reset register	EALLOW	Go
8Ah	SOFTPRES4	Peripheral Software Reset register	EALLOW	Go
8Eh	SOFTPRES6	Peripheral Software Reset register	EALLOW	Go
90h	SOFTPRES7	Peripheral Software Reset register	EALLOW	Go
92h	SOFTPRES8	Peripheral Software Reset register	EALLOW	Go
94h	SOFTPRES9	Peripheral Software Reset register	EALLOW	Go
98h	SOFTPRES11	Peripheral Software Reset register	EALLOW	Go
9Ch	SOFTPRES13	Peripheral Software Reset register	EALLOW	Go

Table 2-80. DEV_CFG_REGS Registers (continued)

Offset	Acronym	Register Name	Write Protection	Section
9Eh	SOFTPRES14	Peripheral Software Reset register	EALLOW	Go
A2h	SOFTPRES16	Peripheral Software Reset register	EALLOW	Go
D6h	CPUSEL0	CPU Select register for common peripherals	EALLOW	Go
D8h	CPUSEL1	CPU Select register for common peripherals	EALLOW	Go
DAh	CPUSEL2	CPU Select register for common peripherals	EALLOW	Go
DCh	CPUSEL3	CPU Select register for common peripherals	EALLOW	Go
DEh	CPUSEL4	CPU Select register for common peripherals	EALLOW	Go
E0h	CPUSEL5	CPU Select register for common peripherals	EALLOW	Go
E2h	CPUSEL6	CPU Select register for common peripherals	EALLOW	Go
E4h	CPUSEL7	CPU Select register for common peripherals	EALLOW	Go
E6h	CPUSEL8	CPU Select register for common peripherals	EALLOW	Go
E8h	CPUSEL9	CPU Select register for common peripherals	EALLOW	Go
ECh	CPUSEL11	CPU Select register for common peripherals	EALLOW	Go
EEh	CPUSEL12	CPU Select register for common peripherals	EALLOW	Go
F2h	CPUSEL14	CPU Select register for common peripherals	EALLOW	Go
122h	CPU2RESCTL	CPU2 Reset Control Register	EALLOW	Go
124h	RSTSTAT	Reset Status register for secondary C28x CPUs		Go
125h	LPMSTAT	LPM Status Register for secondary C28x CPUs		Go

2.14.8.1 DEVCFGLOCK1 Register (Offset = 0h) [reset = X]

DEVCFGLOCK1 is shown in [Figure 2-80](#) and described in [Table 2-81](#).

Lock bit for CPUSELx registers

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

Note:

Any SOnce 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

Figure 2-80. DEVCFGLOCK1 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	CPUSEL14	CPUSEL13	CPUSEL12	CPUSEL11	CPUSEL10	CPUSEL9	CPUSEL8
R=0-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h
7	6	5	4	3	2	1	0
CPUSEL7	CPUSEL6	CPUSEL5	CPUSEL4	CPUSEL3	CPUSEL2	CPUSEL1	CPUSEL0
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-81. DEVCFGLOCK1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R=0	0h	Reserved
14	CPUSEL14	R/SOnce	0h	Lock bit for CPUSEL14 register: 0: Register is not locked 1: Register is locked
13	CPUSEL13	R/SOnce	0h	Lock bit for CPUSEL13 register: 0: Register is not locked 1: Register is locked
12	CPUSEL12	R/SOnce	0h	Lock bit for CPUSEL12 register: 0: Register is not locked 1: Register is locked
11	CPUSEL11	R/SOnce	0h	Lock bit for CPUSEL11 register: 0: Register is not locked 1: Register is locked
10	CPUSEL10	R/SOnce	0h	Lock bit for CPUSEL10 register: 0: Register is not locked 1: Register is locked
9	CPUSEL9	R/SOnce	0h	Lock bit for CPUSEL9 register: 0: Register is not locked 1: Register is locked
8	CPUSEL8	R/SOnce	0h	Lock bit for CPUSEL8 register: 0: Register is not locked 1: Register is locked

Table 2-81. DEVCFGLOCK1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	CPUSEL7	R/SONce	0h	Lock bit for CPUSEL7 register: 0: Register is not locked 1: Register is locked
6	CPUSEL6	R/SONce	0h	Lock bit for CPUSEL6 register: 0: Register is not locked 1: Register is locked
5	CPUSEL5	R/SONce	0h	Lock bit for CPUSEL5 register: 0: Register is not locked 1: Register is locked
4	CPUSEL4	R/SONce	0h	Lock bit for CPUSEL4 register: 0: Register is not locked 1: Register is locked
3	CPUSEL3	R/SONce	0h	Lock bit for CPUSEL3 register: 0: Register is not locked 1: Register is locked
2	CPUSEL2	R/SONce	0h	Lock bit for CPUSEL2 register: 0: Register is not locked 1: Register is locked
1	CPUSEL1	R/SONce	0h	Lock bit for CPUSEL1 register: 0: Register is not locked 1: Register is locked
0	CPUSEL0	R/SONce	X	Lock bit for CPUSEL0 register: 0: Register is not locked 1: Register is locked

2.14.8.2 PARTIDL Register (Offset = 8h) [reset = X]

PARTIDL is shown in [Figure 2-81](#) and described in [Table 2-82](#).

Lower 32-bit of Device PART Identification Number

Figure 2-81. PARTIDL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-82. PARTIDL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	PARTID_FORMAT_REVISION	R	X	Revision of the PARTID format
27-24	RESERVED	R/WOnce	X	Reserved
23-16	FLASH_SIZE	R	X	0x7 - 512KB 0x6 - 256KB Note: This field shows flash size on CPU1 (see datasheet for flash size available)
15	RESERVED	R	X	Reserved
14-13	INSTASPIN	R	X	0 = Reserved for future 1 = Reserved for future 2 = Reserved for future 3 = NONE
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10-8	PIN_COUNT	R	X	0 = reserved for future 1 = reserved for future 2 = reserved for future 3 = reserved for future 4 = reserved for future 5 = 100 pin 6 = 176 pin 7 = 337 pin
7-6	QUAL	R	X	0 = Engineering sample.(TMX) 1 = Pilot production (TMP) 2 = Fully qualified (TMS)
5	RESERVED	R	X	Reserved
4-3	RESERVED	R	0h	Reserved
2-0	RESERVED	R	0h	Reserved

2.14.8.3 PARTIDH Register (Offset = Ah) [reset = X]

PARTIDH is shown in [Figure 2-82](#) and described in [Table 2-83](#).

Upper 32-bit of Device PART Identification Number

Figure 2-82. PARTIDH Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-83. PARTIDH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	DEVICE_CLASS_ID	R	X	Reserved
23-16	PARTNO	R	X	Refer to Datasheet for Device Part Number
15-8	FAMILY	R	X	Device Family 0x3 - DELFINO DUAL CORE 0x4 - DELFINO SINGLE CORE 0x5 - PICCOLO SINGLE CORE Other values Reserved
7-0	RESERVED	R	X	Reserved

2.14.8.4 REVID Register (Offset = Ch) [reset = X]

REVID is shown in [Figure 2-83](#) and described in [Table 2-84](#).

Device Revision Number

Figure 2-83. 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
REVID																															
R-X																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-84. REVID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REVID	R	X	These 32-bits specify the silicon revision 0x0 - Revision 0 0x0 - Revision A

2.14.8.5 DC0 Register (Offset = 10h) [reset = X]

DC0 is shown in [Figure 2-84](#) and described in [Table 2-85](#).

Device Capability: Device Information

Figure 2-84. DC0 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							SINGLE_CORE
R=0-0h							R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-85. DC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-1	RESERVED	R=0	0h	Reserved
0	SINGLE_CORE	R	X	Single Core vs Dual Core 0: Single Core 1: Dual Core

2.14.8.6 DC1 Register (Offset = 12h) [reset = X]

DC1 is shown in [Figure 2-85](#) and described in [Table 2-86](#).

Device Capability: Processing Block Customization

Figure 2-85. DC1 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	CPU2_CLA1
R=0-0h						R-0h	R-X
7	6	5	4	3	2	1	0
RESERVED	CPU1_CLA1	RESERVED		CPU2_VCU	CPU1_VCU	CPU2_FPU_T MU	CPU1_FPU_T MU
R-0h	R-X	R=0-0h		R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-86. DC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-10	RESERVED	R=0	0h	Reserved
9	RESERVED	R	0h	Reserved
8	CPU2_CLA1	R	X	0 - feature is not present on this device 1 - feature is present on this device
7	RESERVED	R	0h	Reserved
6	CPU1_CLA1	R	X	0 - feature is not present on this device 1 - feature is present on this device
5-4	RESERVED	R=0	0h	Reserved
3	CPU2_VCU	R	X	0 - feature is not present on this device 1 - feature is present on this device
2	CPU1_VCU	R	X	0 - feature is not present on this device 1 - feature is present on this device
1	CPU2_FPU_TMU	R	X	0 - feature is not present on this device 1 - feature is present on this device
0	CPU1_FPU_TMU	R	X	0 - feature is not present on this device 1 - feature is present on this device

2.14.8.7 DC2 Register (Offset = 14h) [reset = X]

DC2 is shown in [Figure 2-86](#) and described in [Table 2-87](#).

Device Capability: EMIF Customization

Figure 2-86. DC2 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						EMIF2	EMIF1
R=0-0h						R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-87. DC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	EMIF2	R	X	EMIF2 : 0: Feature not present on the device 1: Feature present on the device
0	EMIF1	R	X	EMIF1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.8 DC3 Register (Offset = 16h) [reset = X]

DC3 is shown in [Figure 2-87](#) and described in [Table 2-88](#).

Device Capability: Peripheral Customization

Figure 2-87. DC3 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	EPWM12	EPWM11	EPWM10	EPWM9
R-0h	R-0h	R-0h	R-0h	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
EPWM8	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-88. DC3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	EPWM12	R	X	EPWM12 : 0: Feature not present on the device 1: Feature present on the device
10	EPWM11	R	X	EPWM11 : 0: Feature not present on the device 1: Feature present on the device
9	EPWM10	R	X	EPWM10 : 0: Feature not present on the device 1: Feature present on the device
8	EPWM9	R	X	EPWM9 : 0: Feature not present on the device 1: Feature present on the device
7	EPWM8	R	X	EPWM8 : 0: Feature not present on the device 1: Feature present on the device
6	EPWM7	R	X	EPWM7 : 0: Feature not present on the device 1: Feature present on the device
5	EPWM6	R	X	EPWM6 : 0: Feature not present on the device 1: Feature present on the device

Table 2-88. DC3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	EPWM5	R	X	EPWM5 : 0: Feature not present on the device 1: Feature present on the device
3	EPWM4	R	X	EPWM4 : 0: Feature not present on the device 1: Feature present on the device
2	EPWM3	R	X	EPWM3 : 0: Feature not present on the device 1: Feature present on the device
1	EPWM2	R	X	EPWM2 : 0: Feature not present on the device 1: Feature present on the device
0	EPWM1	R	X	EPWM1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.9 DC4 Register (Offset = 18h) [reset = X]

DC4 is shown in [Figure 2-88](#) and described in [Table 2-89](#).

Device Capability: Peripheral Customization

Figure 2-88. DC4 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	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
R-0h	R-0h	R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-89. DC4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	ECAP6	R	X	ECAP6 : 0: Feature not present on the device 1: Feature present on the device
4	ECAP5	R	X	ECAP5 : 0: Feature not present on the device 1: Feature present on the device
3	ECAP4	R	X	ECAP4 : 0: Feature not present on the device 1: Feature present on the device
2	ECAP3	R	X	ECAP3 : 0: Feature not present on the device 1: Feature present on the device
1	ECAP2	R	X	ECAP2 : 0: Feature not present on the device 1: Feature present on the device
0	ECAP1	R	X	ECAP1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.10 DC5 Register (Offset = 1Ah) [reset = X]

DC5 is shown in [Figure 2-89](#) and described in [Table 2-90](#).

Device Capability: Peripheral Customization

Figure 2-89. DC5 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	EQEP3	EQEP2	EQEP1
R=0-0h				R=0h	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-90. DC5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	EQEP3	R	X	EQEP3 : 0: Feature not present on the device 1: Feature present on the device
1	EQEP2	R	X	EQEP2 : 0: Feature not present on the device 1: Feature present on the device
0	EQEP1	R	X	EQEP1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.11 DC6 Register (Offset = 1Ch) [reset = X]

DC6 is shown in [Figure 2-90](#) and described in [Table 2-91](#).

Device Capability: Peripheral Customization

Figure 2-90. DC6 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	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-91. DC6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.8.12 DC7 Register (Offset = 1Eh) [reset = X]

DC7 is shown in [Figure 2-91](#) and described in [Table 2-92](#).

Device Capability: Peripheral Customization

Figure 2-91. DC7 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	SD2	SD1
R=0-0h								R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-92. DC7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	SD2	R	X	SD2 : 0: Feature not present on the device 1: Feature present on the device
0	SD1	R	X	SD1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.13 DC8 Register (Offset = 20h) [reset = X]

DC8 is shown in [Figure 2-92](#) and described in [Table 2-93](#).

Device Capability: Peripheral Customization

Figure 2-92. DC8 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				SCI_D	SCI_C	SCI_B	SCI_A
R=0-0h				R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-93. DC8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	SCI_D	R	X	SCI_D : 0: Feature not present on the device 1: Feature present on the device
2	SCI_C	R	X	SCI_C : 0: Feature not present on the device 1: Feature present on the device
1	SCI_B	R	X	SCI_B : 0: Feature not present on the device 1: Feature present on the device
0	SCI_A	R	X	SCI_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.14 DC9 Register (Offset = 22h) [reset = X]

DC9 is shown in [Figure 2-93](#) and described in [Table 2-94](#).

Device Capability: Peripheral Customization

Figure 2-93. DC9 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	SPI_C	SPI_B	SPI_A
R=0-0h				R-0h	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-94. DC9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	SPI_C	R	X	SPI_C : 0: Feature not present on the device 1: Feature present on the device
1	SPI_B	R	X	SPI_B : 0: Feature not present on the device 1: Feature present on the device
0	SPI_A	R	X	SPI_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.15 DC10 Register (Offset = 24h) [reset = X]

DC10 is shown in [Figure 2-94](#) and described in [Table 2-95](#).

Device Capability: Peripheral Customization

Figure 2-94. DC10 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R=0-0h						R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-95. DC10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	I2C_B	R	X	I2C_B : 0: Feature not present on the device 1: Feature present on the device
0	I2C_A	R	X	I2C_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.16 DC11 Register (Offset = 26h) [reset = X]

DC11 is shown in [Figure 2-95](#) and described in [Table 2-96](#).

Device Capability: Peripheral Customization

Figure 2-95. DC11 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-0h	R-0h	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-96. DC11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	CAN_B	R	X	CAN_B : 0: Feature not present on the device 1: Feature present on the device
0	CAN_A	R	X	CAN_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.17 DC12 Register (Offset = 28h) [reset = X]

DC12 is shown in [Figure 2-96](#) and described in [Table 2-97](#).

Device Capability: Peripheral Customization

Figure 2-96. DC12 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-97. DC12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19-18	RESERVED	R	0h	Reserved
17-16	USB_A	R	X	Capability of the USB_A Module: 2'b00: No USB function 2'b01: Device Only 2'b10: Device or Host 2'b11: OTG
15-2	RESERVED	R=0	0h	Reserved
1	McBSP_B	R	X	McBSP_B : 0: Feature not present on the device 1: Feature present on the device
0	McBSP_A	R	X	McBSP_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.18 DC13 Register (Offset = 2Ah) [reset = X]

DC13 is shown in [Figure 2-97](#) and described in [Table 2-98](#).

Device Capability: Peripheral Customization

Figure 2-97. DC13 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	uPP_A
R=0-0h						R-0h	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-98. DC13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	RESERVED	R	0h	Reserved
0	uPP_A	R	X	uPP_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.19 DC14 Register (Offset = 2Ch) [reset = X]

DC14 is shown in [Figure 2-98](#) and described in [Table 2-99](#).

Device Capability: Analog Modules Customization

Figure 2-98. DC14 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				ADC_D	ADC_C	ADC_B	ADC_A
R=0-0h				R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-99. DC14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	ADC_D	R	X	ADC_D : 0: Feature not present on the device 1: Feature present on the device
2	ADC_C	R	X	ADC_C : 0: Feature not present on the device 1: Feature present on the device
1	ADC_B	R	X	ADC_B : 0: Feature not present on the device 1: Feature present on the device
0	ADC_A	R	X	ADC_A : 0: Feature not present on the device 1: Feature present on the device

2.14.8.20 DC15 Register (Offset = 2Eh) [reset = X]

DC15 is shown in [Figure 2-99](#) and described in [Table 2-100](#).

Device Capability: Analog Modules Customization

Figure 2-99. DC15 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
CMPSS8	CMPSS7	CMPSS6	CMPSS5	CMPSS4	CMPSS3	CMPSS2	CMPSS1
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-100. DC15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	CMPSS8	R	X	CMPSS8 : 0: Feature not present on the device 1: Feature present on the device
6	CMPSS7	R	X	CMPSS7 : 0: Feature not present on the device 1: Feature present on the device
5	CMPSS6	R	X	CMPSS6 : 0: Feature not present on the device 1: Feature present on the device
4	CMPSS5	R	X	CMPSS5 : 0: Feature not present on the device 1: Feature present on the device
3	CMPSS4	R	X	CMPSS4 : 0: Feature not present on the device 1: Feature present on the device
2	CMPSS3	R	X	CMPSS3 : 0: Feature not present on the device 1: Feature present on the device
1	CMPSS2	R	X	CMPSS2 : 0: Feature not present on the device 1: Feature present on the device
0	CMPSS1	R	X	CMPSS1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.21 DC17 Register (Offset = 32h) [reset = X]

DC17 is shown in [Figure 2-100](#) and described in [Table 2-101](#).

Device Capability: Analog Modules Customization

Figure 2-100. DC17 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	DAC_C	DAC_B	DAC_A
R=0-0h				R-0h	R-X	R-X	R-X
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				R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-101. DC17 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19	RESERVED	R	0h	Reserved
18	DAC_C	R	X	Buffered-DAC_C : 0: Feature not present on the device 1: Feature present on the device
17	DAC_B	R	X	Buffered-DAC_B : 0: Feature not present on the device 1: Feature present on the device
16	DAC_A	R	X	Buffered-DAC_A : 0: Feature not present on the device 1: Feature present on the device
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.8.22 DC18 Register (Offset = 34h) [reset = X]

DC18 is shown in [Figure 2-101](#) and described in [Table 2-102](#).

Device Capability: CPU1 Lx SRAM Customization

Figure 2-101. DC18 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		LS5_1	LS4_1	LS3_1	LS2_1	LS1_1	LS0_1
R=0-0h		R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-102. DC18 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	RESERVED	R=0	0h	Reserved
5	LS5_1	R	X	LS5_1 : 0: Feature not present on the device 1: Feature present on the device
4	LS4_1	R	X	LS4_1 : 0: Feature not present on the device 1: Feature present on the device
3	LS3_1	R	X	LS3_1 : 0: Feature not present on the device 1: Feature present on the device
2	LS2_1	R	X	LS2_1 : 0: Feature not present on the device 1: Feature present on the device
1	LS1_1	R	X	LS1_1 : 0: Feature not present on the device 1: Feature present on the device
0	LS0_1	R	X	LS0_1 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.23 DC19 Register (Offset = 36h) [reset = X]

DC19 is shown in [Figure 2-102](#) and described in [Table 2-103](#).

Device Capability: CPU2 Lx SRAM Customization

Figure 2-102. DC19 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		LS5_2	LS4_2	LS3_2	LS2_2	LS1_2	LS0_2
R=0-0h		R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-103. DC19 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	RESERVED	R=0	0h	Reserved
5	LS5_2	R	X	LS5_2 : 0: Feature not present on the device 1: Feature present on the device
4	LS4_2	R	X	LS4_2 : 0: Feature not present on the device 1: Feature present on the device
3	LS3_2	R	X	LS3_2 : 0: Feature not present on the device 1: Feature present on the device
2	LS2_2	R	X	LS2_2 : 0: Feature not present on the device 1: Feature present on the device
1	LS1_2	R	X	LS1_2 : 0: Feature not present on the device 1: Feature present on the device
0	LS0_2	R	X	LS0_2 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.24 DC20 Register (Offset = 38h) [reset = X]

DC20 is shown in [Figure 2-103](#) and described in [Table 2-104](#).

Device Capability: GSx SRAM Customization

Figure 2-103. DC20 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
GS15	GS14	GS13	GS12	GS11	GS10	GS9	GS8
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
GS7	GS6	GS5	GS4	GS3	GS2	GS1	GS0
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-104. DC20 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	GS15	R	X	GS15 : 0: Feature not present on the device 1: Feature present on the device
14	GS14	R	X	GS14 : 0: Feature not present on the device 1: Feature present on the device
13	GS13	R	X	GS13 : 0: Feature not present on the device 1: Feature present on the device
12	GS12	R	X	GS12 : 0: Feature not present on the device 1: Feature present on the device
11	GS11	R	X	GS11 : 0: Feature not present on the device 1: Feature present on the device
10	GS10	R	X	GS10 : 0: Feature not present on the device 1: Feature present on the device
9	GS9	R	X	GS9 : 0: Feature not present on the device 1: Feature present on the device
8	GS8	R	X	GS8 : 0: Feature not present on the device 1: Feature present on the device
7	GS7	R	X	GS7 : 0: Feature not present on the device 1: Feature present on the device

Table 2-104. DC20 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	GS6	R	X	GS6 : 0: Feature not present on the device 1: Feature present on the device
5	GS5	R	X	GS5 : 0: Feature not present on the device 1: Feature present on the device
4	GS4	R	X	GS4 : 0: Feature not present on the device 1: Feature present on the device
3	GS3	R	X	GS3 : 0: Feature not present on the device 1: Feature present on the device
2	GS2	R	X	GS2 : 0: Feature not present on the device 1: Feature present on the device
1	GS1	R	X	GS1 : 0: Feature not present on the device 1: Feature present on the device
0	GS0	R	X	GS0 : 0: Feature not present on the device 1: Feature present on the device

2.14.8.25 PERCNF1 Register (Offset = 60h) [reset = X]

PERCNF1 is shown in [Figure 2-104](#) and described in [Table 2-105](#).

Peripheral Configuration register

Figure 2-104. PERCNF1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-105. PERCNF1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	USB_A_PHY	R	X	Internal PHY is present present or not for the USB_A module: 0: Internal USB PHY Module is not present 1: Internal USB PHY Module is present.
15-4	RESERVED	R=0	0h	Reserved
3	ADC_D_MODE	R	X	0: 16-bit or 12-bit configurable in software 1: Only 12-bit operation available
2	ADC_C_MODE	R	X	0: 16-bit or 12-bit configurable in software 1: Only 12-bit operation available
1	ADC_B_MODE	R	X	0: 16-bit or 12-bit configurable in software 1: Only 12-bit operation available
0	ADC_A_MODE	R	X	0: 16-bit or 12-bit configurable in software 1: Only 12-bit operation available

2.14.8.26 FUSEERR Register (Offset = 74h) [reset = 0h]

FUSEERR is shown in [Figure 2-105](#) and described in [Table 2-106](#).

e-Fuse error Status register

Figure 2-105. 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				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-106. FUSEERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	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
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

2.14.8.27 SOFTPRES0 Register (Offset = 82h) [reset = 0h]

SOFTPRES0 is shown in [Figure 2-106](#) and described in [Table 2-107](#).

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 2-106. 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	CPU2_CLA1	RESERVED	CPU1_CLA1
R=0-0h				R-0h	R/W-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-107. SOFTPRES0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	CPU2_CLA1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	RESERVED	R	0h	Reserved
0	CPU1_CLA1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.28 SOFTPRES1 Register (Offset = 84h) [reset = 0h]

SOFTPRES1 is shown in [Figure 2-107](#) and described in [Table 2-108](#).

EMIF Software Reset register

Figure 2-107. SOFTPRES1 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						EMIF2	EMIF1
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-108. SOFTPRES1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	EMIF2	R/W	0h	When this bit is set, only the control logic of the respective EMIF2 is reset. It does not reset the internal registers except the Total Access register and the Total Activate register. This bit must be manually cleared after being set. 1: EMIF2 is under SOFTRESET 0: Module reset is determined by the device Reset Network
0	EMIF1	R/W	0h	When this bit is set, only the control logic of the respective EMIF1 is reset. It does not reset the internal registers except the Total Access register and the Total Activate register. This bit must be manually cleared after being set. 1: EMIF1 is under SOFTRESET 0: Module reset is determined by the device Reset Network

2.14.8.29 SOFTPRES2 Register (Offset = 86h) [reset = 0h]

SOFTPRES2 is shown in [Figure 2-108](#) and described in [Table 2-109](#).

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 2-108. 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	EPWM12	EPWM11	EPWM10	EPWM9
R-0h	R-0h	R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-109. SOFTPRES2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	EPWM12	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
10	EPWM11	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
9	EPWM10	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
8	EPWM9	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
7	EPWM8	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
6	EPWM7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
5	EPWM6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
4	EPWM5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
3	EPWM4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
2	EPWM3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

Table 2-109. SOFTPRES2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	EPWM2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	EPWM1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.30 SOFTPRES3 Register (Offset = 88h) [reset = 0h]

SOFTPRES3 is shown in [Figure 2-109](#) and described in [Table 2-110](#).

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 2-109. 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	RESERVED	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-110. SOFTPRES3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	ECAP6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
4	ECAP5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
3	ECAP4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
2	ECAP3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	ECAP2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	ECAP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.31 SOFTPRES4 Register (Offset = 8Ah) [reset = 0h]

SOFTPRES4 is shown in [Figure 2-110](#) and described in [Table 2-111](#).

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 2-110. 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	EQEP3	EQEP2	EQEP1
R=0-0h				R=0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-111. SOFTPRES4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	EQEP3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	EQEP2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	EQEP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.32 SOFTPRES6 Register (Offset = 8Eh) [reset = 0h]

SOFTPRES6 is shown in [Figure 2-111](#) and described in [Table 2-112](#).

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 2-111. 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	SD2	SD1
R=0-0h								R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W- 0h	R/W- 0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-112. SOFTPRES6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	SD2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	SD1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.33 SOFTPRES7 Register (Offset = 90h) [reset = 0h]

SOFTPRES7 is shown in [Figure 2-112](#) and described in [Table 2-113](#).

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 2-112. 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				SCI_D	SCI_C	SCI_B	SCI_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-113. SOFTPRES7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	SCI_D	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
2	SCI_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	SCI_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	SCI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.34 SOFTPRES8 Register (Offset = 92h) [reset = 0h]

SOFTPRES8 is shown in [Figure 2-113](#) and described in [Table 2-114](#).

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 2-113. 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	SPI_C	SPI_B	SPI_A
R=0-0h				R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-114. SOFTPRES8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	SPI_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	SPI_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	SPI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.35 SOFTPRES9 Register (Offset = 94h) [reset = 0h]

SOFTPRES9 is shown in [Figure 2-114](#) and described in [Table 2-115](#).

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 2-114. SOFTPRES9 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-115. SOFTPRES9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	I2C_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	I2C_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.36 SOFTPRES11 Register (Offset = 98h) [reset = 0h]

SOFTPRES11 is shown in [Figure 2-115](#) and described in [Table 2-116](#).

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 2-115. SOFTPRES11 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED						RESERVED	USB_A
R=0-0h						R-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						McBSP_B	McBSP_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-116. SOFTPRES11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	USB_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
15-2	RESERVED	R=0	0h	Reserved
1	McBSP_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	McBSP_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.37 SOFTPRES13 Register (Offset = 9Ch) [reset = 0h]

SOFTPRES13 is shown in [Figure 2-116](#) and described in [Table 2-117](#).

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 2-116. 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				ADC_D	ADC_C	ADC_B	ADC_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-117. SOFTPRES13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	ADC_D	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
2	ADC_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	ADC_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	ADC_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.38 SOFTPRES14 Register (Offset = 9Eh) [reset = 0h]

SOFTPRES14 is shown in [Figure 2-117](#) and described in [Table 2-118](#).

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 2-117. 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
CMPSS8	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	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-118. SOFTPRES14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	CMPSS8	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
6	CMPSS7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
5	CMPSS6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
4	CMPSS5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
3	CMPSS4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
2	CMPSS3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
1	CMPSS2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
0	CMPSS1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure

2.14.8.39 SOFTPRES16 Register (Offset = A2h) [reset = 0h]

SOFTPRES16 is shown in [Figure 2-118](#) and described in [Table 2-119](#).

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 2-118. SOFTPRES16 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	DAC_C	DAC_B	DAC_A
R=0-0h				R-0h	R/W-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				R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-119. SOFTPRES16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19	RESERVED	R	0h	Reserved
18	DAC_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
17	DAC_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
16	DAC_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.8.40 CPUSEL0 Register (Offset = D6h) [reset = 0h]

CPUSEL0 is shown in [Figure 2-119](#) and described in [Table 2-120](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-119. CPUSEL0 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	EPWM12	EPWM11	EPWM10	EPWM9
R-0h	R-0h	R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-120. CPUSEL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	EPWM12	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
10	EPWM11	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
9	EPWM10	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
8	EPWM9	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
7	EPWM8	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
6	EPWM7	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
5	EPWM6	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
4	EPWM5	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

Table 2-120. CPUSEL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	EPWM4	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
2	EPWM3	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	EPWM2	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	EPWM1	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.41 CPUSEL1 Register (Offset = D8h) [reset = 0h]

CPUSEL1 is shown in [Figure 2-120](#) and described in [Table 2-121](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-120. CPUSEL1 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	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-121. CPUSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	ECAP6	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
4	ECAP5	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
3	ECAP4	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
2	ECAP3	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	ECAP2	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	ECAP1	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.42 CPUSEL2 Register (Offset = DAh) [reset = 0h]

CPUSEL2 is shown in [Figure 2-121](#) and described in [Table 2-122](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-121. CPUSEL2 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	EQEP3	EQEP2	EQEP1
R=0-0h				R=0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-122. CPUSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	EQEP3	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	EQEP2	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	EQEP1	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.43 CPUSEL3 Register (Offset = DCh) [reset = 0h]

CPUSEL3 is shown in [Figure 2-122](#) and described in [Table 2-123](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-122. CPUSEL3 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	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-123. CPUSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.8.44 CPUSEL4 Register (Offset = DEh) [reset = 0h]

CPUSEL4 is shown in [Figure 2-123](#) and described in [Table 2-124](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-123. CPUSEL4 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	SD2	SD1
R=0-0h								R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W- 0h	R/W- 0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-124. CPUSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	SD2	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	SD1	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.45 CPUSEL5 Register (Offset = E0h) [reset = 0h]

CPUSEL5 is shown in [Figure 2-124](#) and described in [Table 2-125](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-124. CPUSEL5 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				SCI_D	SCI_C	SCI_B	SCI_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-125. CPUSEL5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	SCI_D	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
2	SCI_C	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	SCI_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	SCI_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.46 CPUSEL6 Register (Offset = E2h) [reset = 0h]

CPUSEL6 is shown in [Figure 2-125](#) and described in [Table 2-126](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-125. CPUSEL6 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	SPI_C	SPI_B	SPI_A
R=0-0h				R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-126. CPUSEL6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	SPI_C	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	SPI_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	SPI_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.47 CPUSEL7 Register (Offset = E4h) [reset = 0h]

CPUSEL7 is shown in [Figure 2-126](#) and described in [Table 2-127](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-126. CPUSEL7 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-127. CPUSEL7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	I2C_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	I2C_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.48 CPUSEL8 Register (Offset = E6h) [reset = 0h]

CPUSEL8 is shown in [Figure 2-127](#) and described in [Table 2-128](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-127. CPUSEL8 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-0h	R-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-128. CPUSEL8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	CAN_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	CAN_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.49 CPUSEL9 Register (Offset = E8h) [reset = 0h]

CPUSEL9 is shown in [Figure 2-128](#) and described in [Table 2-129](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-128. CPUSEL9 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						McBSP_B	McBSP_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-129. CPUSEL9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	McBSP_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	McBSP_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.50 CPUSEL11 Register (Offset = ECh) [reset = 0h]

CPUSEL11 is shown in [Figure 2-129](#) and described in [Table 2-130](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-129. CPUSEL11 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				ADC_D	ADC_C	ADC_B	ADC_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-130. CPUSEL11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	ADC_D	R/W	0h	0: Connected to CPU1 1: Connected to CPU2 Note: [1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.
2	ADC_C	R/W	0h	0: Connected to CPU1 1: Connected to CPU2 Note: [1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.
1	ADC_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2 Note: [1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.

Table 2-130. CPUSEL11 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ADC_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2 Note: [1] These CPUSEL bits affect the ownership of only ADC Configuration registers by CPU1 or CPU2 (which are mapped on the mapped to VBUS32). ADC result registers are readable from all masters without any CPUSEL dependency.

2.14.8.51 CPUSEL12 Register (Offset = EEh) [reset = 0h]

CPUSEL12 is shown in [Figure 2-130](#) and described in [Table 2-131](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-130. CPUSEL12 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
CMPSS8	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	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-131. CPUSEL12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	CMPSS8	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
6	CMPSS7	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
5	CMPSS6	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
4	CMPSS5	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
3	CMPSS4	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
2	CMPSS3	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
1	CMPSS2	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
0	CMPSS1	R/W	0h	0: Connected to CPU1 1: Connected to CPU2

2.14.8.52 CPUSEL14 Register (Offset = F2h) [reset = 0h]

CPUSEL14 is shown in [Figure 2-131](#) and described in [Table 2-132](#).

CPU Select register for common peripherals

This register must be configured prior to enabling the peripheral clocks.

The clock for each peripheral is derived from the selected CPU subsystem. The clock mux controlled by this register is not glitch-free, therefore the CPUSELx register must be configured before the PCLKCRx register.

The reset for each peripheral is also driven from the selected CPU.

Figure 2-131. CPUSEL14 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	DAC_C	DAC_B	DAC_A
R=0-0h				R-0h	R/W-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				R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-132. CPUSEL14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19	RESERVED	R	0h	Reserved
18	DAC_C	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
17	DAC_B	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
16	DAC_A	R/W	0h	0: Connected to CPU1 1: Connected to CPU2
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.8.53 CPU2RESCTL Register (Offset = 122h) [reset = 1h]

CPU2RESCTL is shown in [Figure 2-132](#) and described in [Table 2-133](#).

CPU2 Reset Control Register

Figure 2-132. CPU2RESCTL 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							RESET
R=0-0h							R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-133. CPU2RESCTL 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
15-1	RESERVED	R=0	0h	Reserved
0	RESET	R/W	1h	This bit controls the reset input of CPU2 core. 1: CPU2 is held in reset (CPU2.RSn = 0) 0: CPU2 reset is deactivated (CPU2.RSn = 1) Note: [1] If CPU2 is not used at-all by an application, it's advisable to put CPU2 in STANDBY mode rather than in reset to save on active power component on the CPU2 subsystem. This is because, all clocks keep toggling when reset is active on the CPU2 sub-system.

2.14.8.54 RSTSTAT Register (Offset = 124h) [reset = 0h]

RSTSTAT is shown in [Figure 2-133](#) and described in [Table 2-134](#).

Reset Status register for secondary C28x CPUs

Figure 2-133. RSTSTAT Register

15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED				CPU2HWBISTRST1	CPU2HWBISTRST0	CPU2NMIWDRST	CPU2RES
R=0-0h				R/W=1-0h	R/W=1-0h	R/W=1-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-134. RSTSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R=0	0h	Reserved
3	CPU2HWBISTRST1	R/W=1	0h	CPU2HWBISTRST0 and CPU2HWBISTRST1 together indicates whether a HWBIST reset was issued to CPU2 or not 00: CPU2 was not reset by the CPU2 HWBIST 11: CPU2 was reset due to CPU2 HWBIST reset This status bit is a latched flag. This flag can be cleared by the CPU1 by writing a 1
2	CPU2HWBISTRST0	R/W=1	0h	CPU2HWBISTRST0 and CPU2HWBISTRST1 together indicates whether a HWBIST reset was issued to CPU2 or not 00: CPU2 was not reset by the CPU2 HWBIST 11: CPU2 was reset due to CPU2 HWBIST reset This status bit is a latched flag. This flag can be cleared by the CPU1 by writing a 1
1	CPU2NMIWDRST	R/W=1	0h	Indicates whether a CPU2.NMIWD reset was issued to CPU2 or not 0: CPU2 was not reset by the CPU2.NMIWD 1: CPU2 was reset due to CPU2.NMIWD reset This status bit is a latched flag. This flag can be cleared by the CPU1 by writing a 1
0	CPU2RES	R	0h	Reset status of CPU2 to CPU1 0: CPU2 core is in reset 1: CPU2 core is out of reset

2.14.8.55 LPMSTAT Register (Offset = 125h) [reset = 0h]

LPMSTAT is shown in [Figure 2-134](#) and described in [Table 2-135](#).

LPM Status Register for secondary C28x CPUs

Figure 2-134. LPMSTAT Register

15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED						CPU2LPMSTAT	
R=0-0h						R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-135. LPMSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R=0	0h	Reserved
1-0	CPU2LPMSTAT	R	0h	These bits indicate the power mode CPU2 00: CPU2 is in ACTIVE mode 01: CPU2 is in IDLE mode 10: CPU2 is in STANDBY mode 11: Reserved

2.14.9 CLK_CFG_REGS Registers

Table 2-136 lists the memory-mapped registers for the CLK_CFG_REGS. All register offset addresses not listed in Table 2-136 should be considered as reserved locations and the register contents should not be modified.

Table 2-136. CLK_CFG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	CLKSEM	Clock Control Semaphore Register	EALLOW	Go
2h	CLKCFGLOCK1	Lock bit for CLKCFG registers	EALLOW	Go
8h	CLKSRCCTL1	Clock Source Control register-1	EALLOW	Go
Ah	CLKSRCCTL2	Clock Source Control register-2	EALLOW	Go
Ch	CLKSRCCTL3	Clock Source Control register-3	EALLOW	Go
Eh	SYSPLLCTL1	SYSPLL Control register-1	EALLOW	Go
14h	SYSPLLMULT	SYSPLL Multiplier register	EALLOW	Go
16h	SYSPLLSTS	SYSPLL Status register		Go
18h	AUXPLLCTL1	AUXPLL Control register-1	EALLOW	Go
1Eh	AUXPLLMULT	AUXPLL Multiplier register	EALLOW	Go
20h	AUXPLLSTS	AUXPLL Status register		Go
22h	SYSCLKDIVSEL	System Clock Divider Select register	EALLOW	Go
24h	AUXCLKDIVSEL	Auxillary Clock Divider Select register	EALLOW	Go
26h	PERCLKDIVSEL	Peripheral Clock Divider Selet register	EALLOW	Go
28h	XCLKOUTDIVSEL	XCLKOUT Divider Select register	EALLOW	Go
2Ch	LOSPCP	Low Speed Clock Source Prescaler	EALLOW	Go
2Eh	MCDCR	Missing Clock Detect Control Register	EALLOW	Go
30h	X1CNT	10-bit Counter on X1 Clock		Go

2.14.9.1 CLKSEM Register (Offset = 0h) [reset = 0h]

CLKSEM is shown in [Figure 2-135](#) and described in [Table 2-137](#).

Clock Control Semaphore Register

Figure 2-135. CLKSEM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-137. CLKSEM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	R=0/W	0h	Writing the value 0xa5a5 will allow the writing of the SEM bits, else writes are ignored. Reads will return 0. 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
15-2	RESERVED	R=0	0h	Reserved
1-0	SEM	R/W	0h	This register provides a mechanism to acquire all the CLKCFG registers (except this register) by CPU1 or CPU2. A CPU can perform read/writes to any of the CLKCFG registers (except this register) only if it owns the semaphore. Otherwise, writes are ignored and reads will return 0x0. Semaphore State Transitions: A value of 00, 10, 11 gives ownership to CPU1 A value of 01 gives ownership to CPU2. The following are the only state transitions allowed on these bits. 00,11 <-> 01 (allowed by CPU2) 00,11 <-> 10 (allowed by CPU1) If a CPU doesn't own the CLK_CFG_REGS set of registers (as defined by the state of this semaphore), reads from that CPU to all those registers return 0x0 and writes are ignore. Note that this is not true of CLKSEM register. The CLKSEM register's reads and writes are always allowed from both CPU1 and CPU2.

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

CLKCFGLOCK1 is shown in [Figure 2-136](#) and described in [Table 2-138](#).

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 2-136. CLKCFGLOCK1 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
LOSPCP	RESERVED	PERCLKDIVSEL	AUXCLKDIVSEL	SYSCLKDIVSEL	AUXPLLMULT	RESERVED	RESERVED
R/SOnce-0h	R=0-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R=0-0h	R=0-0h
7	6	5	4	3	2	1	0
AUXPLLCTL1	SYSPLLMULT	SYSPLLCTL3	SYSPLLCTL2	SYSPLLCTL1	CLKSRCCTL3	CLKSRCCTL2	CLKSRCCTL1
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-138. CLKCFGLOCK1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	LOSPCP	R/SOnce	0h	Lock bit for LOSPCP register: 0: Respective register is not locked 1: Respective register is locked.
14	RESERVED	R=0	0h	Reserved
13	PERCLKDIVSEL	R/SOnce	0h	Lock bit for PERCLKDIVSEL register: 0: Respective register is not locked 1: Respective register is locked.
12	AUXCLKDIVSEL	R/SOnce	0h	Lock bit for AUXCLKDIVSEL register: 0: Respective register is not locked 1: Respective register is locked.
11	SYSCLKDIVSEL	R/SOnce	0h	Lock bit for SYSCLKDIVSEL register: 0: Respective register is not locked 1: Respective register is locked.
10	AUXPLLMULT	R/SOnce	0h	Lock bit for AUXPLLMULT register: 0: Respective register is not locked 1: Respective register is locked.
9	RESERVED	R=0	0h	Reserved
8	RESERVED	R=0	0h	Reserved
7	AUXPLLCTL1	R/SOnce	0h	Lock bit for AUXPLLCTL1 register: 0: Respective register is not locked 1: Respective register is locked.

Table 2-138. CLKCFGLOCK1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	SYSPLLMULT	R/SONce	0h	Lock bit for SYSPLLMULT register: 0: Respective register is not locked 1: Respective register is locked.
5	SYSPLLCTL3	R/SONce	0h	Lock bit for SYSPLLCTL3 register: 0: Respective register is not locked 1: Respective register is locked.
4	SYSPLLCTL2	R/SONce	0h	Lock bit for SYSPLLCTL2 register: 0: Respective register is not locked 1: Respective register is locked.
3	SYSPLLCTL1	R/SONce	0h	Lock bit for SYSPLLCTL1 register: 0: Respective register is not locked 1: Respective register is locked.
2	CLKSRCCTL3	R/SONce	0h	Lock bit for CLKSRCCTL3 register: 0: Respective register is not locked 1: Respective register is locked.
1	CLKSRCCTL2	R/SONce	0h	Lock bit for CLKSRCCTL2 register: 0: Respective register is not locked 1: Respective register is locked.
0	CLKSRCCTL1	R/SONce	0h	Lock bit for CLKSRCCTL1 register: 0: Respective register is not locked 1: Respective register is locked.

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

CLKSRCCTL1 is shown in [Figure 2-137](#) and described in [Table 2-139](#).

Clock Source Control register-1

Figure 2-137. 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	XTALOFF	INTOSC2OFF	RESERVED	OSCCLKSRCSEL	
R=0-0h		R/W-0h	R/W-0h	R/W-0h	R=0-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-139. CLKSRCCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	RESERVED	R=0	0h	Reserved
5	WDHALTI	R/W	0h	<p>Watchdog HALT Mode Ignore Bit: This bit determines if CPU1.WD is functional in the HALT mode or not.</p> <p>0 = CPU1.WD is not functional in the HALT mode. Clock to CPU1.WD is gated when system enters HALT mode. Additionally, INTOSC1 and INTOSC2 are powered-down when system enters HALT mode</p> <p>1 = CPU1.WD is functional in the HALT mode. Clock to CPU1.WD is not gated and INTOSC1/2 are not powered-down when system enters HALT mode</p> <p>Notes:</p> <p>[1] Clock to CPU2.WD clocks is always gated in the HALT mode.</p>
4	XTALOFF	R/W	0h	<p>Crystal (External) Oscillator Off Bit: This bit turns external oscillator off:</p> <p>0 = Crystal (External) Oscillator On (default on reset)</p> <p>1 = Crystal (External) Oscillator Off</p> <p>NOTE: Ensure no resources are using a clock source prior to disabling it. For example OSCCLKSRCSEL (SYSPLL), AUXOSCCLKSRCSEL (AUXPLL), CANxBCLKSEL (CAN Clock), TMR2CLKSRCSEL (CPUTIMER2) and XCLKOUTSEL(XCLKOUT).</p>
3	INTOSC2OFF	R/W	0h	<p>Internal Oscillator 2 Off Bit: This bit turns oscillator 2 off:</p> <p>0 = Internal Oscillator 2 On (default on reset)</p> <p>1 = Internal Oscillator 2 Off</p> <p>This bit could be used by the user to turn off the internal oscillator 2 if it is not used.</p> <p>NOTE: Ensure no resources are using a clock source prior to disabling it. For example OSCCLKSRCSEL (SYSPLL), AUXOSCCLKSRCSEL (AUXPLL), TMR2CLKSRCSEL (CPUTIMER2) and XCLOCKOUT (XCLKOUT).</p>
2	RESERVED	R=0	0h	Reserved

Table 2-139. 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>

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

CLKSRCCTL2 is shown in [Figure 2-138](#) and described in [Table 2-140](#).

Clock Source Control register-2

Figure 2-138. 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						R=0h	
7	6	5	4	3	2	1	0
RESERVED		CANBBCLKSEL		CANABCLKSEL		AUXOSCCLKSRCSEL	
R-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-140. CLKSRCCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-10	RESERVED	R=0	0h	Reserved
9-8	RESERVED	R	0h	Reserved
7-6	RESERVED	R	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 = AUXCLKIN (from GPIO) 11 = Reserved Missing clock detect circuit doesnt have any impact on these bits.
3-2	CANABCLKSEL	R/W	0h	CANA Bit-Clock Source Select Bit: 00 = PERx.SYSCLK (default on reset) 01 = External Oscillator (XTAL) 10 = AUXCLKIN (from GPIO) 11 = Reserved Missing clock detect circuit doesnt have any impact on these bits.

Table 2-140. CLKSRCCTL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	AUXOSCCLKSRCSEL	R/W	0h	<p>Oscillator Clock Source Select Bit: This bit selects the source for AUXOSCCLK:</p> <p>00 = INTOSC2 (default on reset)</p> <p>01 = External Oscillator (XTAL)</p> <p>10 = AUXCLKIN (from GPIO)</p> <p>11 = Reserved</p> <p>Whenever the user changes the clock source using these bits, the AUXPLLMULT 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 AUXPLLMULT register to configure the appropriate multiplier.</p> <p>The user must wait 10 OSCCLK cycles before writing to AUXPLLMULT</p> <p>or disabling the previous clock source to allow the change to complete.</p> <p>The missing clock detection circuit does not affect these bits.</p>

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

CLKSRCCTL3 is shown in [Figure 2-139](#) and described in [Table 2-141](#).

Clock Source Control register-3

Figure 2-139. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-141. CLKSRCCTL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-3	RESERVED	R=0	0h	Reserved
2-0	XCLKOUTSEL	R/W	0h	<p>XCLKOUT Source Select Bit: This bit selects the source for XCLKOUT:</p> <p>000 = PLLSYSCLK (default on reset)</p> <p>001 = PLLRAWCLK</p> <p>010 = CPU1.SYSCLK</p> <p>011 = CPU2.SYSCLK</p> <p>100 = AUXPLLRAWCLK</p> <p>101 = INTOSC1</p> <p>110 = INTOSC2</p> <p>111 = Reserved</p>

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

SYSPLLCTL1 is shown in [Figure 2-140](#) and described in [Table 2-142](#).

SYSPLL Control register-1

Figure 2-140. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-142. SYSPLLCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	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
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

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

SYSPLLMULT is shown in [Figure 2-141](#) and described in [Table 2-143](#).

SYSPLL Multiplier register

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

Figure 2-141. SYSPLLMULT 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						FMULT	
R=0-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		IMULT					
R=0-0h		R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-143. SYSPLLMULT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
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
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

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

SYSPLLSTS is shown in [Figure 2-142](#) and described in [Table 2-144](#).

SYSPLL Status register

Figure 2-142. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-144. SYSPLLSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	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 Note: [1] If SYSPLL out of lock condition is detected then interrupts are fired to CPU1 and CPU2 through their respective ePIE modules. Software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler
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

2.14.9.9 AUXPLLCTL1 Register (Offset = 18h) [reset = 0h]

AUXPLLCTL1 is shown in [Figure 2-143](#) and described in [Table 2-145](#).

AUXPLL Control register-1

Figure 2-143. AUXPLLCTL1 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-145. AUXPLLCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	PLLCLKEN	R/W	0h	AUXPLL bypassed or included in the AUXPLLCLK path: This bit decides if the AUXPLL is bypassed when AUXPLLCLK is generated 1 = AUXPLLCLK is fed from the AUXPLL clock output. Users need to make sure that the PLL is locked before enabling this clock to the AUXPLLCLK connected modules. 0 = AUXPLL is bypassed. Clock to modules connected to AUXPLLCLK is direct feed from AUXOSCCLK
0	PLLEN	R/W	0h	AUXPLL enabled or disabled: This bit decides if the AUXPLL is enabled or not 1 = AUXPLL is enabled 0 = AUXPLL is powered off. Clock to system is direct feed from AUXOSCCLK

2.14.9.10 AUXPLLMULT Register (Offset = 1Eh) [reset = 0h]

AUXPLLMULT is shown in [Figure 2-144](#) and described in [Table 2-146](#).

AUXPLL Multiplier register

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

Figure 2-144. AUXPLLMULT 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						FMULT	
R=0-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		IMULT					
R=0-0h		R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-146. AUXPLLMULT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-10	RESERVED	R=0	0h	Reserved
9-8	FMULT	R/W	0h	AUXPLL Fractional Multiplier : 00 Fractional Multiplier = 0 01 Fractional Multiplier = 0.25 10 Fractional Multiplier = 0.5 11 Fractional Multiplier = 0.75
7	RESERVED	R=0	0h	Reserved
6-0	IMULT	R/W	0h	AUXPLL 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

2.14.9.11 AUXPLLSTS Register (Offset = 20h) [reset = 0h]

AUXPLLSTS is shown in [Figure 2-145](#) and described in [Table 2-147](#).

AUXPLL Status register

Figure 2-145. AUXPLLSTS 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-147. AUXPLLSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	SLIPS	R	0h	AUXPLL Slip Status Bit: This bit indicates whether the AUXPLL is out of lock range 0 = AUXPLL is not out of lock 1 = AUXPLL is out of lock Note: [1] If AUXPLL out of lock condition is detected then interrupts are fired to CPU1 and CPU2 through their respective ePIE modules. Software can decide to relock the PLL or switch to PLL bypass mode in the interrupt handler
0	LOCKS	R	0h	AUXPLL Lock Status Bit: This bit indicates whether the AUXPLL is locked or not 0 = AUXPLL is not yet locked 1 = AUXPLL is locked

2.14.9.12 SYSCLKDIVSEL Register (Offset = 22h) [reset = 2h]

SYSCLKDIVSEL is shown in [Figure 2-146](#) and described in [Table 2-148](#).

System Clock Divider Select register

Figure 2-146. SYSCLKDIVSEL 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					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-148. SYSCLKDIVSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	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

2.14.9.13 AUXCLKDIVSEL Register (Offset = 24h) [reset = 1h]

AUXCLKDIVSEL is shown in [Figure 2-147](#) and described in [Table 2-149](#).

Auxillary Clock Divider Select register

Figure 2-147. AUXCLKDIVSEL 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						AUXPLLDIV	
R=0-0h						R/W-1h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-149. AUXCLKDIVSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1-0	AUXPLLDIV	R/W	1h	AUXPLLCLK Divide Select: This bit selects the divider setting for the AUXPLLCK. 00 = /1 01 = /2 (default on reset) 10 = /4 11 = /8

2.14.9.14 PERCLKDIVSEL Register (Offset = 26h) [reset = 51h]

PERCLKDIVSEL is shown in [Figure 2-148](#) and described in [Table 2-150](#).

Peripheral Clock Divider Selet register

Figure 2-148. PERCLKDIVSEL 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	EMIF2CLKDIV	RESERVED	EMIF1CLKDIV	RESERVED	EPWMCLKDIV		
R=0-0h	R/W-1h	R=0-0h	R/W-1h	R=0-0h	R/W-1h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-150. PERCLKDIVSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-7	RESERVED	R=0	0h	Reserved
6	EMIF2CLKDIV	R/W	1h	EMIF2 Clock Divide Select: This bit selects whether the EMIF2 module run with a /1 or /2 clock. 0: /1 of CPU1.SYSCLK is selected 1: /2 of CPU1.SYSCLK is selected
5	RESERVED	R=0	0h	Reserved
4	EMIF1CLKDIV	R/W	1h	EMIF1 Clock Divide Select: This bit selects whether the EMIF1 module run with a /1 or /2 clock. For single core device 0: /1 of CPU1.SYSCLK is selected 1: /2 of CPU1.SYSCLK is selected For Dual core device 0: /1 of PLLSYSCLK is selected 1: /2 of PLLSYSCLK is selected
3-2	RESERVED	R	0h	Reserved
1-0	EPWMCLKDIV	R/W	1h	EPWM Clock Divide Select: This bit selects whether the EPWM modules run with a /1 or /2 clock. This divider sits in front of the PLLSYSCLK x0 = /1 of PLLSYSCLK x1 = /2 of PLLSYSCLK (default on reset) Note: Refer to the EPWM User Guide for maximum EPWM Frequency

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

XCLKOUTDIVSEL is shown in [Figure 2-149](#) and described in [Table 2-151](#).

XCLKOUT Divider Select register

Figure 2-149. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-151. XCLKOUTDIVSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	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)

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

LOSPCP is shown in [Figure 2-150](#) and described in [Table 2-152](#).

Low Speed Clock Source Prescaler

Figure 2-150. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-152. LOSPCP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-3	RESERVED	R=0	0h	Reserved
2-0	LSPCLKDIV	R/W	2h	<p>These bits configure the low-speed peripheral clock (LSPCLK) rate relative to SYSCLK of CPU1 and CPU2.</p> <p>000,LSPCLK = / 1</p> <p>001,LSPCLK = / 2</p> <p>010,LSPCLK = / 4 (default on reset)</p> <p>011,LSPCLK = / 6</p> <p>100,LSPCLK = / 8</p> <p>101,LSPCLK = / 10</p> <p>110,LSPCLK = / 12</p> <p>111,LSPCLK = / 14</p> <p>Note:</p> <p>[1] This clock is used as strobe for the SCI and SPI modules.</p>

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

MCDCR is shown in [Figure 2-151](#) and described in [Table 2-153](#).

Missing Clock Detect Control Register

Figure 2-151. 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/W=1-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-153. MCDCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	OSCOFF	R/W	0h	Oscillator Clock Off Bit: 0 = OSCCLK Connected to OSCCLK Counter in MCD module 1 = OSCCLK Disconnected to OSCCLK Counter in MCD module
2	MCLKOFF	R/W	0h	Missing Clock Detect Off Bit: 0 = Missing Clock Detect Circuit Enabled 1 = Missing Clock Detect Circuit Disabled
1	MCLKCLR	R=0/W=1	0h	Missing Clock Clear Bit: Write 1" to this bit to clear MCLKSTS bit and reset the missing clock detect circuit."
0	MCLKSTS	R	0h	Missing Clock Status Bit: 0 = OSCCLK Is OK 1 = OSCCLK Detected Missing, CLOCKFAILn Generated

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

X1CNT is shown in [Figure 2-152](#) and described in [Table 2-154](#).

10-bit Counter on X1 Clock

Figure 2-152. X1CNT 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								X1CNT							
R=0-0h								R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-154. X1CNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-10	RESERVED	R=0	0h	Reserved
9-0	X1CNT	R	0h	X1 Counter: <ul style="list-style-type: none"> - 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 guarantee that the Crystal connected to X1/X2 is powered Up.

2.14.10 CPU_SYS_REGS Registers

Table 2-155 lists the memory-mapped registers for the CPU_SYS_REGS. All register offset addresses not listed in Table 2-155 should be considered as reserved locations and the register contents should not be modified.

Table 2-155. CPU_SYS_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	CPUSYSLOCK1	Lock bit for CPUSYS registers	EALLOW	Go
6h	HIBBOOTMODE	HIB Boot Mode Register	EALLOW	Go
8h	IORESTOREADDR	IORestore() routine Address Register	EALLOW	Go
Ah	PIEVERRADDR	PIE Vector Fetch Error Address register	EALLOW	Go
22h	PCLKCR0	Peripheral Clock Gating Registers	EALLOW	Go
24h	PCLKCR1	Peripheral Clock Gating Registers	EALLOW	Go
26h	PCLKCR2	Peripheral Clock Gating Registers	EALLOW	Go
28h	PCLKCR3	Peripheral Clock Gating Registers	EALLOW	Go
2Ah	PCLKCR4	Peripheral Clock Gating Registers	EALLOW	Go
2Eh	PCLKCR6	Peripheral Clock Gating Registers	EALLOW	Go
30h	PCLKCR7	Peripheral Clock Gating Registers	EALLOW	Go
32h	PCLKCR8	Peripheral Clock Gating Registers	EALLOW	Go
34h	PCLKCR9	Peripheral Clock Gating Registers	EALLOW	Go
36h	PCLKCR10	Peripheral Clock Gating Registers	EALLOW	Go
38h	PCLKCR11	Peripheral Clock Gating Registers	EALLOW	Go
3Ah	PCLKCR12	Peripheral Clock Gating Registers	EALLOW	Go
3Ch	PCLKCR13	Peripheral Clock Gating Registers	EALLOW	Go
3Eh	PCLKCR14	Peripheral Clock Gating Registers	EALLOW	Go
42h	PCLKCR16	Peripheral Clock Gating Registers	EALLOW	Go
74h	SECMSEL	Secondary Master Select register for common peripherals: Selects between CLA & DMA	EALLOW	Go
76h	LPMCR	LPM Control Register	EALLOW	Go
78h	GPIOLPMSEL0	GPIO LPM Wakeup select registers	EALLOW	Go
7Ah	GPIOLPMSEL1	GPIO LPM Wakeup select registers	EALLOW	Go
7Ch	TMR2CLKCTL	Timer2 Clock Measurement functionality control register	EALLOW	Go
80h	RESC	Reset Cause register		Go

2.14.10.1 CPUSYSLOCK1 Register (Offset = 0h) [reset = 0h]

CPUSYSLOCK1 is shown in [Figure 2-153](#) and described in [Table 2-156](#).

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 2-153. CPUSYSLOCK1 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
GPIOLPMSEL1	GPIOLPMSEL0	LPMCR	SECMSEL	PCLKCR16	PCLKCR15	PCLKCR14	PCLKCR13
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h
15	14	13	12	11	10	9	8
PCLKCR12	PCLKCR11	PCLKCR10	PCLKCR9	PCLKCR8	PCLKCR7	PCLKCR6	PCLKCR5
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h
7	6	5	4	3	2	1	0
PCLKCR4	PCLKCR3	PCLKCR2	PCLKCR1	PCLKCR0	PIEVERRADD R	IORESTOREA DDR	HIBBOOTMOD E
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-156. CPUSYSLOCK1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R=0	0h	Reserved
23	GPIOLPMSEL1	R/SOnce	0h	Lock bit for GPIOLPMSEL1 Register: 0: Respective register is not locked 1: Respective register is locked.
22	GPIOLPMSEL0	R/SOnce	0h	Lock bit for GPIOLPMSEL0 Register: 0: Respective register is not locked 1: Respective register is locked.
21	LPMCR	R/SOnce	0h	Lock bit for LPMCR Register: 0: Respective register is not locked 1: Respective register is locked.
20	SECMSEL	R/SOnce	0h	Lock bit for SECMSEL Register: 0: Respective register is not locked 1: Respective register is locked.
19	PCLKCR16	R/SOnce	0h	Lock bit for PCLKCR16 Register: 0: Respective register is not locked 1: Respective register is locked.
18	PCLKCR15	R/SOnce	0h	Lock bit for PCLKCR15 Register: 0: Respective register is not locked 1: Respective register is locked.
17	PCLKCR14	R/SOnce	0h	Lock bit for PCLKCR14 Register: 0: Respective register is not locked 1: Respective register is locked.

Table 2-156. CPUSYSLOCK1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
16	PCLKCR13	R/SONce	0h	Lock bit for PCLKCR13 Register: 0: Respective register is not locked 1: Respective register is locked.
15	PCLKCR12	R/SONce	0h	Lock bit for PCLKCR12 Register: 0: Respective register is not locked 1: Respective register is locked.
14	PCLKCR11	R/SONce	0h	Lock bit for PCLKCR11 Register: 0: Respective register is not locked 1: Respective register is locked.
13	PCLKCR10	R/SONce	0h	Lock bit for PCLKCR10 Register: 0: Respective register is not locked 1: Respective register is locked.
12	PCLKCR9	R/SONce	0h	Lock bit for PCLKCR9 Register: 0: Respective register is not locked 1: Respective register is locked.
11	PCLKCR8	R/SONce	0h	Lock bit for PCLKCR8 Register: 0: Respective register is not locked 1: Respective register is locked.
10	PCLKCR7	R/SONce	0h	Lock bit for PCLKCR7 Register: 0: Respective register is not locked 1: Respective register is locked.
9	PCLKCR6	R/SONce	0h	Lock bit for PCLKCR6 Register: 0: Respective register is not locked 1: Respective register is locked.
8	PCLKCR5	R/SONce	0h	Lock bit for PCLKCR5 Register: 0: Respective register is not locked 1: Respective register is locked.
7	PCLKCR4	R/SONce	0h	Lock bit for PCLKCR4 Register: 0: Respective register is not locked 1: Respective register is locked.
6	PCLKCR3	R/SONce	0h	Lock bit for PCLKCR3 Register: 0: Respective register is not locked 1: Respective register is locked.
5	PCLKCR2	R/SONce	0h	Lock bit for PCLKCR2 Register: 0: Respective register is not locked 1: Respective register is locked.
4	PCLKCR1	R/SONce	0h	Lock bit for PCLKCR1 Register: 0: Respective register is not locked 1: Respective register is locked.
3	PCLKCR0	R/SONce	0h	Lock bit for PCLKCR0 Register: 0: Respective register is not locked 1: Respective register is locked.

Table 2-156. CPUSYSLOCK1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	PIEVERRADDR	R/SOnce	0h	Lock bit for PIEVERRADDR Register: 0: Respective register is not locked 1: Respective register is locked.
1	IORESTOREADDR	R/SOnce	0h	Lock bit for IORESTOREADDR Register: 0: Respective register is not locked 1: Respective register is locked.
0	HIBBOOTMODE	R/SOnce	0h	Lock bit for HIBBOOTMODE register: 0: Respective register is not locked 1: Respective register is locked.

2.14.10.2 HIBBOOTMODE Register (Offset = 6h) [reset = Fh]

HIBBOOTMODE is shown in [Figure 2-154](#) and described in [Table 2-157](#).

HIB Boot Mode Register

Figure 2-154. HIBBOOTMODE Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-157. HIBBOOTMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BMODE	R/W	Fh	This register defined the boot mode on a HIB Wakeup. Its the responsibility of user to initialize the appropriate boot mode before going into HIB mode. Refer to the Boot ROM section for more details on this register

2.14.10.3 IORESTOREADDR Register (Offset = 8h) [reset = 3FFFFFFh]

IORESTOREADDR is shown in [Figure 2-155](#) and described in [Table 2-158](#).

IORestore() routine Address Register

Figure 2-155. IORESTOREADDR Register

31	30	29	28	27	26	25	24	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-3FFFFFFh																				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-158. IORESTOREADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R=0	0h	Reserved
21-0	ADDR	R/W	3FFFFFFh	This register defines the address of the restoreIO() routine on a HIB wakeup. Its the responsibility of user to initialize this register with the restoreIO() routine address before going into HIB mode. Refer to the Boot ROM section for more details on this register.

2.14.10.4 PIEVERRADDR Register (Offset = Ah) [reset = 3FFFFFFh]

PIEVERRADDR is shown in [Figure 2-156](#) and described in [Table 2-159](#).

PIE Vector Fetch Error Address register

Figure 2-156. 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-3FFFFFFh																				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-159. PIEVERRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R=0	0h	Reserved
21-0	ADDR	R/W	3FFFFFFh	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 0x3fffbfe will get executed. Refer to the Boot ROM section for more details on this register.

2.14.10.5 PCLKCR0 Register (Offset = 22h) [reset = 38h]

PCLKCR0 is shown in [Figure 2-157](#) and described in [Table 2-160](#).

Peripheral Clock Gating Registers

Figure 2-157. PCLKCR0 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED				GTBCLKSYNC	TBCLKSYNC	RESERVED	HRPWM
R=0-0h				R/W-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=0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-160. PCLKCR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19	GTBCLKSYNC	R/W	0h	EPWM Time Base Clock Global sync: When set by CPU1, PWM time bases of all modules start counting. The effect of this bit is seen on all the EPMW modules irrespective of their partitioning based on CPUSEL Notes: 1. This bit on the CPU2.PCLKCR0 register has no effect. 2. Writing '1' to this bit overrides the effect of write '1' to the TBCLKSYNC bit at the same time
18	TBCLKSYNC	R/W	0h	EPWM Time Base Clock sync: When set PWM time bases of all the PWM modules belonging to the same CPU-Subsystem (as partitioned using their CPUSEL bits) start counting Notes: 1. This bit from CPU1.PCLKCR0 or CPU2.PCLKCR0 is selected and fed to the individual EPWM modules based on their respective CPUSEL bit.
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 Note: [1] This bit is present only in CPU1.PCLKCR0. This bit is not used (R/W) in CPU2.PCLKCR0
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
4	CPUTIMER1	R/W	1h	CPUTIMER1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

Table 2-160. PCLKCR0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CPUTIMER0	R/W	1h	CPUTIMER0 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	DMA	R/W	0h	DMA Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	RESERVED	R	0h	Reserved
0	CLA1	R/W	0h	CLA1 Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.6 PCLKCR1 Register (Offset = 24h) [reset = 0h]

PCLKCR1 is shown in [Figure 2-158](#) and described in [Table 2-161](#).

Peripheral Clock Gating Registers

Figure 2-158. PCLKCR1 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						EMIF2	EMIF1
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-161. PCLKCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	EMIF2	R/W	0h	EMIF2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register.
0	EMIF1	R/W	0h	EMIF1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register.

2.14.10.7 PCLKCR2 Register (Offset = 26h) [reset = 0h]

PCLKCR2 is shown in [Figure 2-159](#) and described in [Table 2-162](#).

Peripheral Clock Gating Registers

Figure 2-159. 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	EPWM12	EPWM11	EPWM10	EPWM9
R-0h	R-0h	R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-162. PCLKCR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	EPWM12	R/W	0h	EPWM12 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
10	EPWM11	R/W	0h	EPWM11 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
9	EPWM10	R/W	0h	EPWM10 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
8	EPWM9	R/W	0h	EPWM9 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
7	EPWM8	R/W	0h	EPWM8 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
6	EPWM7	R/W	0h	EPWM7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
5	EPWM6	R/W	0h	EPWM6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

Table 2-162. PCLKCR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	EPWM5	R/W	0h	EPWM5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
3	EPWM4	R/W	0h	EPWM4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	EPWM3	R/W	0h	EPWM3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	EPWM2	R/W	0h	EPWM2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	EPWM1	R/W	0h	EPWM1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.8 PCLKCR3 Register (Offset = 28h) [reset = 0h]

PCLKCR3 is shown in [Figure 2-160](#) and described in [Table 2-163](#).

Peripheral Clock Gating Registers

Figure 2-160. 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	RESERVED	ECAP6	ECAP5	ECAP4	ECAP3	ECAP2	ECAP1
R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-163. PCLKCR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	ECAP6	R/W	0h	ECAP6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
4	ECAP5	R/W	0h	ECAP5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
3	ECAP4	R/W	0h	ECAP4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	ECAP3	R/W	0h	ECAP3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	ECAP2	R/W	0h	ECAP2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	ECAP1	R/W	0h	ECAP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.9 PCLKCR4 Register (Offset = 2Ah) [reset = 0h]

PCLKCR4 is shown in [Figure 2-161](#) and described in [Table 2-164](#).

Peripheral Clock Gating Registers

Figure 2-161. 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	EQEP3	EQEP2	EQEP1
R=0-0h				R=0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-164. PCLKCR4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	EQEP3	R/W	0h	EQEP3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	EQEP2	R/W	0h	EQEP2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	EQEP1	R/W	0h	EQEP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.10 PCLKCR6 Register (Offset = 2Eh) [reset = 0h]

PCLKCR6 is shown in [Figure 2-162](#) and described in [Table 2-165](#).

Peripheral Clock Gating Registers

Figure 2-162. 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	SD2	SD1
R=0-0h								R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W- 0h	R/W- 0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-165. PCLKCR6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	SD2	R/W	0h	SD2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	SD1	R/W	0h	SD1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.11 PCLKCR7 Register (Offset = 30h) [reset = 0h]

PCLKCR7 is shown in [Figure 2-163](#) and described in [Table 2-166](#).

Peripheral Clock Gating Registers

Figure 2-163. 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				SCI_D	SCI_C	SCI_B	SCI_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-166. PCLKCR7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	SCI_D	R/W	0h	SCI_D Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	SCI_C	R/W	0h	SCI_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	SCI_B	R/W	0h	SCI_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	SCI_A	R/W	0h	SCI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.12 PCLKCR8 Register (Offset = 32h) [reset = 0h]

PCLKCR8 is shown in [Figure 2-164](#) and described in [Table 2-167](#).

Peripheral Clock Gating Registers

Figure 2-164. 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	SPI_C	SPI_B	SPI_A
R=0-0h				R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-167. PCLKCR8 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	SPI_C	R/W	0h	SPI_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	SPI_B	R/W	0h	SPI_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	SPI_A	R/W	0h	SPI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.13 PCLKCR9 Register (Offset = 34h) [reset = 0h]

PCLKCR9 is shown in [Figure 2-165](#) and described in [Table 2-168](#).

Peripheral Clock Gating Registers

Figure 2-165. PCLKCR9 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						R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-168. PCLKCR9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	I2C_B	R/W	0h	I2C_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	I2C_A	R/W	0h	I2C_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.14 PCLKCR10 Register (Offset = 36h) [reset = 0h]

PCLKCR10 is shown in [Figure 2-166](#) and described in [Table 2-169](#).

Peripheral Clock Gating Registers

Figure 2-166. 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-0h	R-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-169. PCLKCR10 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	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
0	CAN_A	R/W	0h	CAN_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.15 PCLKCR11 Register (Offset = 38h) [reset = 0h]

PCLKCR11 is shown in [Figure 2-167](#) and described in [Table 2-170](#).

Peripheral Clock Gating Registers

Figure 2-167. PCLKCR11 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED						RESERVED	USB_A
R=0-0h						R-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						McBSP_B	McBSP_A
R=0-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-170. PCLKCR11 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R=0	0h	Reserved
17	RESERVED	R	0h	Reserved
16	USB_A	R/W	0h	USB_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] This bit is not used (R/W) in CPU2.PCLKCR11 register. USB_A clock enabled is controlled only from CPU1.PCLKCR11 register
15-2	RESERVED	R=0	0h	Reserved
1	McBSP_B	R/W	0h	McBSP_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	McBSP_A	R/W	0h	McBSP_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.16 PCLKCR12 Register (Offset = 3Ah) [reset = 0h]

PCLKCR12 is shown in [Figure 2-168](#) and described in [Table 2-171](#).

Peripheral Clock Gating Registers

Figure 2-168. PCLKCR12 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	uPP_A
R=0-0h						R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-171. PCLKCR12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	RESERVED	R	0h	Reserved
0	uPP_A	R/W	0h	uPP_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] This bit also affects the uPP message RAM wrapper associated with the respective uPP module [2] This bit is not used (R/W) in CPU2.PCLKCR12 register. UPP_A clock enabled is controlled only from CPU1.PCLKCR12 register

2.14.10.17 PCLKCR13 Register (Offset = 3Ch) [reset = 0h]

PCLKCR13 is shown in [Figure 2-169](#) and described in [Table 2-172](#).

Peripheral Clock Gating Registers

Figure 2-169. 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				ADC_D	ADC_C	ADC_B	ADC_A
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-172. PCLKCR13 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-4	RESERVED	R=0	0h	Reserved
3	ADC_D	R/W	0h	ADC_D Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	ADC_C	R/W	0h	ADC_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	ADC_B	R/W	0h	ADC_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	ADC_A	R/W	0h	ADC_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.18 PCLKCR14 Register (Offset = 3Eh) [reset = 0h]

PCLKCR14 is shown in [Figure 2-170](#) and described in [Table 2-173](#).

Peripheral Clock Gating Registers

Figure 2-170. 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
CMPSS8	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	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-173. PCLKCR14 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-8	RESERVED	R=0	0h	Reserved
7	CMPSS8	R/W	0h	CMPSS8 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
6	CMPSS7	R/W	0h	CMPSS7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
5	CMPSS6	R/W	0h	CMPSS6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
4	CMPSS5	R/W	0h	CMPSS5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
3	CMPSS4	R/W	0h	CMPSS4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
2	CMPSS3	R/W	0h	CMPSS3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
1	CMPSS2	R/W	0h	CMPSS2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on
0	CMPSS1	R/W	0h	CMPSS1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on

2.14.10.19 PCLKCR16 Register (Offset = 42h) [reset = 0h]

PCLKCR16 is shown in [Figure 2-171](#) and described in [Table 2-174](#).

Peripheral Clock Gating Registers

Figure 2-171. PCLKCR16 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	DAC_C	DAC_B	DAC_A
R=0-0h				R-0h	R/W-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				R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-174. PCLKCR16 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R=0	0h	Reserved
19	RESERVED	R	0h	Reserved
18	DAC_C	R/W	0h	Buffered_DAC_C Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on
17	DAC_B	R/W	0h	Buffered_DAC_B Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on
16	DAC_A	R/W	0h	Buffered_DAC_A Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on
15-4	RESERVED	R=0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.10.20 SECMSEL Register (Offset = 74h) [reset = 0h]

SECMSEL is shown in [Figure 2-172](#) and described in [Table 2-175](#).

Secondary Master Select register for common peripherals: Selects between CLA & DMA

Figure 2-172. SECMSEL 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	
R=0-0h		R=0h		R=0h		R=0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		PF2SEL		PF1SEL	
R=0h		R=0h		R/W=0h		R/W=0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-175. SECMSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-14	RESERVED	R=0	0h	Reserved
13-12	RESERVED	R	0h	Reserved
11-10	RESERVED	R	0h	Reserved
9-8	RESERVED	R	0h	Reserved
7-6	RESERVED	R	0h	Reserved
5-4	RESERVED	R	0h	Reserved
3-2	PF2SEL	R/W	0h	This bit selects whether the dual ported bridge is connected with DMA or CLA as the secondary master (C28x is always connected as primary master) x0: Bridge is connected to CLA x1: Bridge is connected to DMA
1-0	PF1SEL	R/W	0h	This bit selects whether the dual ported bridge is connected with DMA or CLA as the secondary master (C28x is always connected as primary master) x0: Bridge is connected to CLA x1: Bridge is connected to DMA

2.14.10.21 LPMCR Register (Offset = 76h) [reset = FCh]

LPMCR is shown in [Figure 2-173](#) and described in [Table 2-176](#).

LPM Control Register

Figure 2-173. LPMCR Register

31	30	29	28	27	26	25	24
IOISODIS	RESERVED						
R/W=1-0h				R=0-0h			
23	22	21	20	19	18	17	16
RESERVED						M0M1MODE	
R=0-0h						R/W-0h	
15	14	13	12	11	10	9	8
WDINTE	RESERVED						
R/W-0h				R=0-0h			
7	6	5	4	3	2	1	0
QUALSTDBY						LPM	
R/W-3Fh						R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-176. LPMCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IOISODIS	R/W=1	0h	0: Indicates IO ISOLATION is not turned ON 1: Indicates IO ISOLATION is turned ON. This bit is set one by hardware ONLY during HIB. This bit cant be set to 1 by software Writing 0 to this bit has not effect. Writing 1 to this bit deactivates IO ISOLATION Notes: [1] This bit is reserved in the register mapped to CPU2
30-18	RESERVED	R=0	0h	Reserved
17-16	M0M1MODE	R/W	0h	These bit control the state of CPU1's and CPU2's M0 & M1 memories when Device goes into HIB mode. 00: CPUx's M0 & M1 memories ON with low-leakage mode 01: CPUx's M0 & M1 memories OFF 1x: Reserved Notes: [1] Low-leakage mode for M0 & M1 memories uses the "Retention" feature of the SRAMs. [2] These bits take effect only when device goes into HIB mode. If the device is not in HIB mode, the value in this bit doesn't control the state of CPU1's and CPU2's M0 & M1 memories
15	WDINTE	R/W	0h	When this bit is set to 1, it enables the watchdog interrupt signal to wake the device from STANDBY mode. Note: [1] To use this signal, the user must also enable the WDINTn signal using the WDENINT bit in the SCSR register. This signal will not wake the device from HALT mode because the clock to watchdog module is turned off
14-8	RESERVED	R=0	0h	Reserved

Table 2-176. LPMCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-2	QUALSTDBY	R/W	3Fh	<p>Select number of OSCCLK clock cycles to qualify the selected inputs when waking the from STANDBY mode:</p> <p>000000 = 2 OSCCLKs</p> <p>000001 = 3 OSCCLKs</p> <p>.....</p> <p>111111 = 65 OSCCLKs</p> <p>Note: The LPMCR.QUALSTDBY register should be set to a value greater than the ratio of INTOSC1/PLLSYSCLK to ensure proper wake up.</p>
1-0	LPM	R/W	0h	<p>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)</p> <p>00: IDLE Mode</p> <p>01: STANDBY Mode</p> <p>10: HALT Mode (treated as STANDBY for CPU2)</p> <p>11: HIB Mode (treated as STANDBY for CPU2)</p> <p>Note:</p> <p>[1] When debugger is connected to CPU1 (CPU1.DCON=1), if HIB is selected, system will enter HALT mode instead if SYSDBGCTL.HIBDIS=1. Refer to SYSDBGCTL for description on this.</p>

2.14.10.22 GPIO LPMSEL0 Register (Offset = 78h) [reset = 0h]

GPIO LPMSEL0 is shown in [Figure 2-174](#) and described in [Table 2-177](#).

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 2-174. GPIO LPMSEL0 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-177. 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
30	GPIO30	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
29	GPIO29	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
28	GPIO28	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
27	GPIO27	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
26	GPIO26	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
25	GPIO25	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
24	GPIO24	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
23	GPIO23	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
22	GPIO22	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
21	GPIO21	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
20	GPIO20	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit

Table 2-177. GPIOLPMSEL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
18	GPIO18	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
17	GPIO17	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
16	GPIO16	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
15	GPIO15	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
14	GPIO14	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
13	GPIO13	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
12	GPIO12	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
11	GPIO11	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
10	GPIO10	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
9	GPIO9	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
8	GPIO8	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
7	GPIO7	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
6	GPIO6	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
5	GPIO5	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
4	GPIO4	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
3	GPIO3	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
2	GPIO2	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
1	GPIO1	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
0	GPIO0	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit

2.14.10.23 GPIO LPMSEL1 Register (Offset = 7Ah) [reset = 0h]

GPIO LPMSEL1 is shown in [Figure 2-175](#) and described in [Table 2-178](#).

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 2-175. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-178. 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
30	GPIO62	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
29	GPIO61	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
28	GPIO60	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
27	GPIO59	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
26	GPIO58	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
25	GPIO57	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
24	GPIO56	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
23	GPIO55	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
22	GPIO54	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
21	GPIO53	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
20	GPIO52	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit

Table 2-178. GPIOLPMSEL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	GPIO51	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
18	GPIO50	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
17	GPIO49	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
16	GPIO48	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
15	GPIO47	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
14	GPIO46	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
13	GPIO45	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
12	GPIO44	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
11	GPIO43	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
10	GPIO42	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
9	GPIO41	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
8	GPIO40	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
7	GPIO39	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
6	GPIO38	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
5	GPIO37	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
4	GPIO36	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
3	GPIO35	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
2	GPIO34	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
1	GPIO33	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit
0	GPIO32	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit

2.14.10.24 TMR2CLKCTL Register (Offset = 7Ch) [reset = 0h]

TMR2CLKCTL is shown in [Figure 2-176](#) and described in [Table 2-179](#).

Timer2 Clock Measurement functionality control register

Figure 2-176. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-179. TMR2CLKCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-6	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>
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 = Reserved</p> <p>101 = Reserved</p> <p>110 = AUXPLLCLK</p> <p>111 = reserved</p>

2.14.10.25 RESC Register (Offset = 80h) [reset = X]

RESC is shown in [Figure 2-177](#) and described in [Table 2-180](#).

Reset Cause register

Figure 2-177. RESC Register

31	30	29	28	27	26	25	24
TRSTn_pin_status	XRSn_pin_status	RESERVED					
R-X	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/W=1-0h
7	6	5	4	3	2	1	0
RESERVED	HIBRESETn	HWBISTn	RESERVED	NMIWDRSn	WDRSn	XRSn	POR
R=0-0h	R/W=1-0h	R/W=1-0h	R=0-0h	R/W=1-0h	R/W=1-0h	R/W=1-1h	R/W=1-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-180. RESC Register Field Descriptions

Bit	Field	Type	Reset	Description
31	TRSTn_pin_status	R	X	Reading this bit provides the current status of TRSTn at the respective C28x CPUs TRSTn input port. Reset value is reflective of the pin status.
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.
29-16	RESERVED	R=0	0h	Reserved
15-9	RESERVED	R=0	0h	Reserved
8	SCCRESETn	R/W=1	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.
7	RESERVED	R=0	0h	Reserved
6	HIBRESETn	R/W=1	0h	If this bit is set, indicates that the device was reset due to a Hibernate mode Wakeup. Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.
5	HWBISTn	R/W=1	0h	If this bit is set, indicates that the device was reset by HWBIST. Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.
4	RESERVED	R=0	0h	Reserved
3	NMIWDRSn	R/W=1	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 CPU1/2.NMISHDFLG registers
2	WDRSn	R/W=1	0h	If this bit is set, indicates that the device was reset by WDRSn. Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.

Table 2-180. RESC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	XRSn	R/W=1	1h	If this bit is set, indicates that the device was reset by XRSn. Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.
0	POR	R/W=1	1h	If this bit is set, indicates that the device was reset by PORn. Writing a 1 to this bit will force the bit to 0 Writing of 0 will have no effect.

2.14.11 ROM_PREFETCH_REGS Registers

[Table 2-181](#) lists the memory-mapped registers for the ROM_PREFETCH_REGS. All register offset addresses not listed in [Table 2-181](#) should be considered as reserved locations and the register contents should not be modified.

Table 2-181. ROM_PREFETCH_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	ROM_PREFETCH	ROM Prefetch Configuration Register	EALLOW	Go

2.14.11.1 ROMPREFETCH Register (Offset = 0h) [reset = 0h]

ROMPREFETCH is shown in [Figure 2-178](#) and described in [Table 2-182](#).

ROM Prefetch Configuration Register

Figure 2-178. ROMPREFETCH 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							PFENABLE
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-182. ROMPREFETCH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	PFENABLE	R/W	0h	0: Prefetch is disabled for secure ROM and boot ROM. 1: Prefetch is enabled for secure ROM and boot ROM.

2.14.12 DCSM_Z1_OTP Registers

Table 2-183 lists the memory-mapped registers for the DCSM_Z1_OTP. All register offset addresses not listed in Table 2-183 should be considered as reserved locations and the register contents should not be modified.

Table 2-183. DCSM_Z1_OTP Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	Z1OTP_LINKPOINTER1	Zone 1 Link Pointer1 in Z1 OTP		Go
4h	Z1OTP_LINKPOINTER2	Zone 1 Link Pointer2 in Z1 OTP		Go
8h	Z1OTP_LINKPOINTER3	Zone 1 Link Pointer3 in Z1 OTP		Go
10h	Z1OTP_PSWDLOCK	Secure Password Lock in Z1 OTP		Go
14h	Z1OTP_CRCLOCK	Secure CRC Lock in Z1 OTP		Go
1Eh	Z1OTP_BOOTCTRL	Boot Mode in Z1 OTP		Go

2.14.12.1 Z1OTP_LINKPOINTER1 Register (Offset = 0h) [reset = FFFFFFFFh]

Z1OTP_LINKPOINTER1 is shown in [Figure 2-179](#) and described in [Table 2-184](#).

Zone 1 Link Pointer1 in Z1 OTP

Figure 2-179. Z1OTP_LINKPOINTER1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_LINKPOINTER1																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-184. Z1OTP_LINKPOINTER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER1	R	FFFFFFFh	Zone1 Link Pointer 1 location in USER OTP.

2.14.12.2 Z1OTP_LINKPOINTER2 Register (Offset = 4h) [reset = FFFFFFFFh]

Z1OTP_LINKPOINTER2 is shown in [Figure 2-180](#) and described in [Table 2-185](#).

Zone 1 Link Pointer2 in Z1 OTP

Figure 2-180. Z1OTP_LINKPOINTER2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_LINKPOINTER2																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-185. Z1OTP_LINKPOINTER2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER2	R	FFFFFFFh	Zone1 Link Pointer 2 location in USER OTP.

2.14.12.3 Z1OTP_LINKPOINTER3 Register (Offset = 8h) [reset = FFFFFFFFh]

Z1OTP_LINKPOINTER3 is shown in [Figure 2-181](#) and described in [Table 2-186](#).

Zone 1 Link Pointer3 in Z1 OTP

Figure 2-181. Z1OTP_LINKPOINTER3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_LINKPOINTER3																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-186. Z1OTP_LINKPOINTER3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER3	R	FFFFFFFh	Zone1 Link Pointer 3 location in USER OTP.

2.14.12.4 Z1OTP_PSWDLOCK Register (Offset = 10h) [reset = FFFFFFFFh]

Z1OTP_PSWDLOCK is shown in [Figure 2-182](#) and described in [Table 2-187](#).

Secure Password Lock in Z1 OTP

Figure 2-182. Z1OTP_PSWDLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_PSWDLOCK																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-187. Z1OTP_PSWDLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_PSWDLOCK	R	FFFFFFFh	Zone1 password lock location in USER OTP.

2.14.12.5 Z1OTP_CRCLOCK Register (Offset = 14h) [reset = FFFFFFFFh]

Z1OTP_CRCLOCK is shown in [Figure 2-183](#) and described in [Table 2-188](#).

Secure CRC Lock in Z1 OTP

Figure 2-183. Z1OTP_CRCLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_CRCLOCK																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-188. Z1OTP_CRCLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_CRCLOCK	R	FFFFFFFh	Zone1 CRC lock location in USER OTP.

2.14.12.6 Z1OTP_BOOTCTRL Register (Offset = 1Eh) [reset = FFFFFFFFh]

Z1OTP_BOOTCTRL is shown in [Figure 2-184](#) and described in [Table 2-189](#).

Boot Mode in Z1 OTP

Figure 2-184. Z1OTP_BOOTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z1OTP_BOOTCTRL																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-189. Z1OTP_BOOTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1OTP_BOOTCTRL	R	FFFFFFFh	Zone1 Boot control location in USER OTP.

2.14.13 DCSM_Z2_OTP Registers

Table 2-190 lists the memory-mapped registers for the DCSM_Z2_OTP. All register offset addresses not listed in Table 2-190 should be considered as reserved locations and the register contents should not be modified.

Table 2-190. DCSM_Z2_OTP Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	Z2OTP_LINKPOINTER1	Zone 2 Link Pointer1 in Z2 OTP		Go
4h	Z2OTP_LINKPOINTER2	Zone 2 Link Pointer2 in Z2 OTP		Go
8h	Z2OTP_LINKPOINTER3	Zone 2 Link Pointer3 in Z2 OTP		Go
10h	Z2OTP_PSWDLOCK	Secure Password Lock in Z2 OTP		Go
14h	Z2OTP_CRCLOCK	Secure CRC Lock in Z2 OTP		Go
1Eh	Z2OTP_BOOTCTRL	Boot Mode in Z2 OTP		Go

2.14.13.1 Z2OTP_LINKPOINTER1 Register (Offset = 0h) [reset = FFFFFFFFh]

Z2OTP_LINKPOINTER1 is shown in [Figure 2-185](#) and described in [Table 2-191](#).

Zone 2 Link Pointer1 in Z2 OTP

Figure 2-185. Z2OTP_LINKPOINTER1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_LINKPOINTER1																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-191. Z2OTP_LINKPOINTER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER1	R	FFFFFFFh	Zone2 Link Pointer 1 location in USER OTP.

2.14.13.2 Z2OTP_LINKPOINTER2 Register (Offset = 4h) [reset = FFFFFFFFh]

Z2OTP_LINKPOINTER2 is shown in [Figure 2-186](#) and described in [Table 2-192](#).

Zone 2 Link Pointer2 in Z2 OTP

Figure 2-186. Z2OTP_LINKPOINTER2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_LINKPOINTER2																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-192. Z2OTP_LINKPOINTER2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER2	R	FFFFFFFh	Zone2 Link Pointer 2 location in USER OTP.

2.14.13.3 Z2OTP_LINKPOINTER3 Register (Offset = 8h) [reset = FFFFFFFFh]

Z2OTP_LINKPOINTER3 is shown in [Figure 2-187](#) and described in [Table 2-193](#).

Zone 2 Link Pointer3 in Z2 OTP

Figure 2-187. Z2OTP_LINKPOINTER3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_LINKPOINTER3																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-193. Z2OTP_LINKPOINTER3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER3	R	FFFFFFFh	Zone2 Link Pointer 3 location in USER OTP.

2.14.13.4 Z2OTP_PSWDLOCK Register (Offset = 10h) [reset = FFFFFFFFh]

Z2OTP_PSWDLOCK is shown in [Figure 2-188](#) and described in [Table 2-194](#).

Secure Password Lock in Z2 OTP

Figure 2-188. Z2OTP_PSWDLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_PSWDLOCK																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-194. Z2OTP_PSWDLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_PSWDLOCK	R	FFFFFFFh	Zone2 password lock location in USER OTP.

2.14.13.5 Z2OTP_CRCLOCK Register (Offset = 14h) [reset = FFFFFFFFh]

Z2OTP_CRCLOCK is shown in [Figure 2-189](#) and described in [Table 2-195](#).

Secure CRC Lock in Z2 OTP

Figure 2-189. Z2OTP_CRCLOCK Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_CRCLOCK																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-195. Z2OTP_CRCLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_CRCLOCK	R	FFFFFFFh	Zone2 CRC lock location in USER OTP.

2.14.13.6 Z2OTP_BOOTCTRL Register (Offset = 1Eh) [reset = FFFFFFFFh]

Z2OTP_BOOTCTRL is shown in [Figure 2-190](#) and described in [Table 2-196](#).

Boot Mode in Z2 OTP

Figure 2-190. Z2OTP_BOOTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z2OTP_BOOTCTRL																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-196. Z2OTP_BOOTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2OTP_BOOTCTRL	R	FFFFFFFh	Zone2 Boot control location in USER OTP.

2.14.14 DCSM_Z1_REGS Registers

Table 2-197 lists the memory-mapped registers for the DCSM_Z1_REGS. All register offset addresses not listed in Table 2-197 should be considered as reserved locations and the register contents should not be modified.

Table 2-197. DCSM_Z1_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	Z1_LINKPOINTER	Zone 1 Link Pointer		Go
2h	Z1_OTPSECLOCK	Zone 1 OTP Secure JTAG lock		Go
4h	Z1_BOOTCTRL	Boot Mode		Go
6h	Z1_LINKPOINTERERR	Link Pointer Error		Go
10h	Z1_CSMKEY0	Zone 1 CSM Key 0		Go
12h	Z1_CSMKEY1	Zone 1 CSM Key 1		Go
14h	Z1_CSMKEY2	Zone 1 CSM Key 2		Go
16h	Z1_CSMKEY3	Zone 1 CSM Key 3		Go
19h	Z1_CR	Zone 1 CSM Control Register		Go
1Ah	Z1_GRABSECTR	Zone 1 Grab Flash Sectors Register		Go
1Ch	Z1_GRABRAMR	Zone 1 Grab RAM Blocks Register		Go
1Eh	Z1_EXEONLYSECTR	Zone 1 Flash Execute_Only Sector Register		Go
20h	Z1_EXEONLYRAMR	Zone 1 RAM Execute_Only Block Register		Go

2.14.14.1 Z1_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

Z1_LINKPOINTER is shown in [Figure 2-191](#) and described in [Table 2-198](#).

Zone 1 Link Pointer

Figure 2-191. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-198. 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 value which is generated by looking at the three physical Link-Pointer values loaded from OTP.

2.14.14.2 Z1_OTPSECLOCK Register (Offset = 2h) [reset = FFFh]

Z1_OTPSECLOCK is shown in [Figure 2-192](#) and described in [Table 2-199](#).

Zone 1 OTP Secure JTAG lock

Figure 2-192. 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-Fh				R-Fh				R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-199. 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	Fh	Value in this field gets loaded from Z1_CRCLOCK[11:8] when a read is issued to address location of Z1_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.
7-4	PSWDLOCK	R	Fh	Value in this field gets loaded from Z1_PSWDLOCK[7:4] when a read is issued to address location of Z1_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.
3-0	RESERVED	R	0h	Reserved

2.14.14.3 Z1_BOOTCTRL Register (Offset = 4h) [reset = 0h]

Z1_BOOTCTRL is shown in [Figure 2-193](#) and described in [Table 2-200](#).

Boot Mode

Figure 2-193. Z1_BOOTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTPIN1								BOOTPIN0								BMODE								KEY							
R-0h								R-0h								R-0h								R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-200. Z1_BOOTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	BOOTPIN1	R	0h	<p>This field gets loaded with Z1_BOOTCTRL[31:24] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.</p> <p>This assigns the pin to be used as BOOTPIN1.</p> <p>0 : Pick default bootmode pin.</p> <p>1 : Pick GPIO0 as BOOTPIN1.</p> <p>2 : Pick GPIO1 as BOOTPIN1.</p> <p>....</p> <p>....</p> <p>n : Pick GPIO_n-1 as BOOTPIN1.</p>
23-16	BOOTPIN0	R	0h	<p>This field gets loaded with Z1_BOOTCTRL[23:16] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.</p> <p>This assigns the pin to be used as BOOTPIN1.</p> <p>0 : Pick default bootmode pin.</p> <p>1 : Pick GPIO0 as BOOTPIN1.</p> <p>2 : Pick GPIO1 as BOOTPIN1.</p> <p>....</p> <p>....</p> <p>n : Pick GPIO_n-1 as BOOTPIN1.</p>
15-8	BMODE	R	0h	<p>This field gets loaded with Z1_BOOTCTRL[16:8] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.</p>
7-0	KEY	R	0h	<p>This field gets loaded with Z1_BOOTCTRL[7:0] when a dummy read is issued to address location of Z1_BOOTCTRL in OTP.</p>

2.14.14.4 Z1_LINKPOINTERERR Register (Offset = 6h) [reset = FFFFFFFFh]

Z1_LINKPOINTERERR is shown in [Figure 2-194](#) and described in [Table 2-201](#).

Link Pointer Error

Figure 2-194. Z1_LINKPOINTERERR Register

31	30	29	28	27	26	25	24	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_LINKPOINTERERR																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-201. Z1_LINKPOINTERERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z1_LINKPOINTERERR	R	FFFFFFFh	<p>These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP.</p> <p>0 : No Error.</p> <p>Other : Error on bit positions which is set to 1.</p>

2.14.14.5 Z1_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z1_CSMKEY0 is shown in [Figure 2-195](#) and described in [Table 2-202](#).

Zone 1 CSM Key 0

Figure 2-195. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-202. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.14.6 Z1_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z1_CSMKEY1 is shown in [Figure 2-196](#) and described in [Table 2-203](#).

Zone 1 CSM Key 1

Figure 2-196. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-203. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.14.7 Z1_CSMKEY2 Register (Offset = 14h) [reset = 0h]

Z1_CSMKEY2 is shown in [Figure 2-197](#) and described in [Table 2-204](#).

Zone 1 CSM Key 2

Figure 2-197. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-204. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.14.8 Z1_CSMKEY3 Register (Offset = 16h) [reset = 0h]

Z1_CSMKEY3 is shown in [Figure 2-198](#) and described in [Table 2-205](#).

Zone 1 CSM Key 3

Figure 2-198. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-205. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.14.9 Z1_CR Register (Offset = 19h) [reset = 8h]

Z1_CR is shown in [Figure 2-199](#) and described in [Table 2-206](#).

Zone 1 CSM Control Register

Figure 2-199. 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=0h	R=1h	R=0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-206. 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.
14-8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	ARMED	R	0h	0 : Dummy read to CSM Password locations in OTP hasn't been performed. 1 : Dummy read to CSM Password locations in OTP has been performed.
5	UNSECURE	R	0h	Indicates the state of Zone. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state.
4	ALLONE	R	0h	Indicates the state of CSM passowrds. 0 : CSM Passwords are not all ones. 1 : CSM Passwords are all ones and zone is in unlock state.
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.
2-0	RESERVED	R	0h	Reserved

2.14.14.10 Z1_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

Z1_GRABSECTR is shown in [Figure 2-200](#) and described in [Table 2-207](#).

Zone 1 Grab Flash Sectors Register

Figure 2-200. Z1_GRABSECTR Register

31	30	29	28	27	26	25	24
RESERVED		GRAB_BANK2		GRAB_SECTN		GRAB_SECTM	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECTL		GRAB_SECTK		GRAB_SECTJ		GRAB_SECTI	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECTH		GRAB_SECTG		GRAB_SECTF		GRAB_SECTE	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECTD		GRAB_SECTC		GRAB_SECTB		GRAB_SECTA	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-207. Z1_GRABSECTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	GRAB_BANK2	R	0h	Value in this field gets loaded from Z1_GRABSECTR[29:28] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash BANK2 is inaccessible. 01 : Request to allocate Flash BANK2 to Zone1. 10 : Request to allocate Flash BANK2 to Zone1. 11 : Request to make Flash BANK2 Non-Secure.
27-26	GRAB_SECTN	R	0h	Value in this field gets loaded from Z1_GRABSECTR[27:26] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector N is inaccessible. 01 : Request to allocate Flash Sector N to Zone1. 10 : Request to allocate Flash Sector N to Zone1. 11 : Request to make Flash sector N Non-Secure.
25-24	GRAB_SECTM	R	0h	Value in this field gets loaded from Z1_GRABSECTR[25:24] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector M is inaccessible. 01 : Request to allocate Flash Sector M to Zone1. 10 : Request to allocate Flash Sector M to Zone1. 11 : Request to make Flash sector M Non-Secure.
23-22	GRAB_SECTL	R	0h	Value in this field gets loaded from Z1_GRABSECTR[23:22] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector L is inaccessible. 01 : Request to allocate Flash Sector L to Zone1. 10 : Request to allocate Flash Sector L to Zone1. 11 : Request to make Flash sector L Non-Secure.

Table 2-207. Z1_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	GRAB_SECTK	R	0h	Value in this field gets loaded from Z1_GRABSECTR[21:20] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector K is inaccessible. 01 : Request to allocate Flash Sector K to Zone1. 10 : Request to allocate Flash Sector K to Zone1. 11 : Request to make Flash sector K Non-Secure.
19-18	GRAB_SECTJ	R	0h	Value in this field gets loaded from Z1_GRABSECTR[19:18] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector J is inaccessible. 01 : Request to allocate Flash Sector J to Zone1. 10 : Request to allocate Flash Sector J to Zone1. 11 : Request to make Flash sector J Non-Secure.
17-16	GRAB_SECTI	R	0h	Value in this field gets loaded from Z1_GRABSECTR[17:16] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector I is inaccessible. 01 : Request to allocate Flash Sector I to Zone1. 10 : Request to allocate Flash Sector I to Zone1. 11 : Request to make Flash sector I Non-Secure.
15-14	GRAB_SECTH	R	0h	Value in this field gets loaded from Z1_GRABSECTR[15:14] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector H is inaccessible. 01 : Request to allocate Flash Sector H to Zone1. 10 : Request to allocate Flash Sector H to Zone1. 11 : Request to make Flash sector H Non-Secure.
13-12	GRAB_SECTG	R	0h	Value in this field gets loaded from Z1_GRABSECTR[13:12] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector G is inaccessible. 01 : Request to allocate Flash Sector G to Zone1. 10 : Request to allocate Flash Sector G to Zone1. 11 : Request to make Flash sector G Non-Secure.
11-10	GRAB_SECTF	R	0h	Value in this field gets loaded from Z1_GRABSECTR[11:10] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector F is inaccessible. 01 : Request to allocate Flash Sector F to Zone1. 10 : Request to allocate Flash Sector F to Zone1. 11 : Request to make Flash sector F Non-Secure.
9-8	GRAB_SECTE	R	0h	Value in this field gets loaded from Z1_GRABSECTR[9:8] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector E is inaccessible. 01 : Request to allocate Flash Sector E to Zone1. 10 : Request to allocate Flash Sector E to Zone1. 11 : Request to make Flash sector E Non-Secure.
7-6	GRAB_SECTD	R	0h	Value in this field gets loaded from Z1_GRABSECTR[7:6] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector D is inaccessible. 01 : Request to allocate Flash Sector D to Zone1. 10 : Request to allocate Flash Sector D to Zone1. 11 : Request to make Flash sector D Non-Secure.

Table 2-207. Z1_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	GRAB_SECTC	R	0h	Value in this field gets loaded from Z1_GRABSECTR[5:4] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector C is inaccessible. 01 : Request to allocate Flash Sector C to Zone1. 10 : Request to allocate Flash Sector C to Zone1. 11 : Request to make Flash sector C Non-Secure.
3-2	GRAB_SECTB	R	0h	Value in this field gets loaded from Z1_GRABSECTR[3:2] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector B is inaccessible. 01 : Request to allocate Flash Sector B to Zone1. 10 : Request to allocate Flash Sector B to Zone1. 11 : Request to make Flash sector B Non-Secure.
1-0	GRAB_SECTA	R	0h	Value in this field gets loaded from Z1_GRABSECTR[1:0] when a read is issued to address location of Z1_GRABSECTR in OTP. 00 : Invalid. Flash Sector A is inaccessible. 01 : Request to allocate Flash Sector A to Zone1. 10 : Request to allocate Flash Sector A to Zone1. 11 : Request to make Flash sector A Non-Secure.

2.14.14.11 Z1_GRABRAMR Register (Offset = 1Ch) [reset = 0h]

Z1_GRABRAMR is shown in [Figure 2-201](#) and described in [Table 2-208](#).

Zone 1 Grab RAM Blocks Register

Figure 2-201. Z1_GRABRAMR Register

31	30	29	28	27	26	25	24
RESERVED		GRAB_CLA1		RESERVED			
R-0h		R-0h		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-208. Z1_GRABRAMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	GRAB_CLA1	R	0h	Value in this field gets loaded from Z1_GRABRAM[29:28] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. CLA1 is inaccessible. 01 : Request to allocate CLA1 to Zone1. 10 : Request to allocate CLA1 to Zone1. 11 : Request to make CLA1 Non-Secure.
27-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from Z1_GRABRAM[15:14] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. D1 RAM is inaccessible. 01 : Request to allocate D1 RAM to Zone1. 10 : Request to allocate D1 RAM to Zone1. 11 : Request to make D1 RAM Non-Secure.
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from Z1_GRABRAM[13:12] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. D0 RAM is inaccessible. 01 : Request to allocate D0 RAM to Zone1. 10 : Request to allocate D0 RAM to Zone1. 11 : Request to make D0 RAM Non-Secure.
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from Z1_GRABRAM[11:10] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS5 RAM is inaccessible. 01 : Request to allocate LS5 RAM to Zone1. 10 : Request to allocate LS5 RAM to Zone1. 11 : Request to make LS5 RAM Non-Secure.

Table 2-208. Z1_GRABRAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	Value in this field gets loaded from Z1_GRABRAM[9:8] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS4 RAM is inaccessible. 01 : Request to allocate LS4 RAM to Zone1. 10 : Request to allocate LS4 RAM to Zone1. 11 : Request to make LS4 RAM Non-Secure.
7-6	GRAB_RAM3	R	0h	Value in this field gets loaded from Z1_GRABRAM[7:6] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS3 RAM is inaccessible. 01 : Request to allocate LS3 RAM to Zone1. 10 : Request to allocate LS3 RAM to Zone1. 11 : Request to make LS3 RAM Non-Secure.
5-4	GRAB_RAM2	R	0h	Value in this field gets loaded from Z1_GRABRAM[5:4] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS2 RAM is inaccessible. 01 : Request to allocate LS2 RAM to Zone1. 10 : Request to allocate LS2 RAM to Zone1. 11 : Request to make LS2 RAM Non-Secure.
3-2	GRAB_RAM1	R	0h	Value in this field gets loaded from Z1_GRABRAM[3:2] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS1 RAM is inaccessible. 01 : Request to allocate LS1 RAM to Zone1. 10 : Request to allocate LS1 RAM to Zone1. 11 : Request to make LS1 RAM Non-Secure.
1-0	GRAB_RAM0	R	0h	Value in this field gets loaded from Z1_GRABRAM[1:0] when a read is issued to address location of Z1_GRABRAM in OTP. 00 : Invalid. LS0 RAM is inaccessible. 01 : Request to allocate LS0 RAM to Zone1. 10 : Request to allocate LS0 RAM to Zone1. 11 : Request to make LS0 RAM Non-Secure.

2.14.14.12 Z1_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

Z1_EXEONLYSECTR is shown in [Figure 2-202](#) and described in [Table 2-209](#).

Zone 1 Flash Execute_Only Sector Register

Figure 2-202. 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
RESERVED	EXEONLY_BA NK2	EXEONLY_SE CTN	EXEONLY_SE CTM	EXEONLY_SE CTL	EXEONLY_SE CTK	EXEONLY_SE CTJ	EXEONLY_SE CTI
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 CTH	EXEONLY_SE CTG	EXEONLY_SE CTF	EXEONLY_SE CTE	EXEONLY_SE CTD	EXEONLY_SE CTC	EXEONLY_SE CTB	EXEONLY_SE CTA
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-209. Z1_EXEONLYSECTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	EXEONLY_BANK2	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[14:14] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash BANK2 (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash BANK2 (only if it's allocated to Zone1)
13	EXEONLY_SECTN	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[13:13] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector N (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector N (only if it's allocated to Zone1)
12	EXEONLY_SECTM	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[12:12] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector M (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector M (only if it's allocated to Zone1)
11	EXEONLY_SECTL	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[11:11] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector L (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector L (only if it's allocated to Zone1)
10	EXEONLY_SECTK	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[10:10] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector K (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector K (only if it's allocated to Zone1)

Table 2-209. Z1_EXEONLYSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	EXEONLY_SECTJ	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[9:9] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector J (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector J (only if it's allocated to Zone1)
8	EXEONLY_SECTI	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[8:8] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector I (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector I (only if it's allocated to Zone1)
7	EXEONLY_SECTH	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[7:7] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector H (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector H (only if it's allocated to Zone1)
6	EXEONLY_SECTG	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[6:6] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector G (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector G (only if it's allocated to Zone1)
5	EXEONLY_SECTF	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[5:5] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector F (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector F (only if it's allocated to Zone1)
4	EXEONLY_SECTE	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[4:4] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector E (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector E (only if it's allocated to Zone1)
3	EXEONLY_SECTD	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[3:3] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector D (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector D (only if it's allocated to Zone1)
2	EXEONLY_SECTC	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[2:2] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector C (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector C (only if it's allocated to Zone1)
1	EXEONLY_SECTB	R	0h	Value in this field gets loaded from Z1_EXEONLYSECTR[1:1] when a read is issued to Z1_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector B (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for Flash Sector B (only if it's allocated to Zone1)

Table 2-209. Z1_EXEONLYSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	EXEONLY_SECTA	R	0h	<p>Value in this field gets loaded from Z1_EXEONLYSECT[0:0] when a read is issued to Z1_EXEONLYSECT address location in OTP.</p> <p>0 : Execute-Only protection is enabled for Flash Sector A (only if it's allocated to Zone1)</p> <p>1 : Execute-Only protection is disabled for Flash Sector A (only if it's allocated to Zone1)</p>

2.14.14.13 Z1_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]

Z1_EXEONLYRAMR is shown in [Figure 2-203](#) and described in [Table 2-210](#).

Zone 1 RAM Execute_Only Block Register

Figure 2-203. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-210. 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 Z1_EXEONLYRAM[7:7] when a read is issued to Z1_EXEONLYRAM address location in OTP. 0 : Execute-Only protection is enabled for D1 RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for D1 RAM (only if it's allocated to Zone1)
6	EXEONLY_RAM6	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[6:6] when a read is issued to Z1_EXEONLYRAM address location in OTP. 0 : Execute-Only protection is enabled for D0 RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for D0 RAM (only if it's allocated to Zone1)
5	EXEONLY_RAM5	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[5:5] when a read is issued to Z1_EXEONLYRAM address location in 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)
4	EXEONLY_RAM4	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[4:4] when a read is issued to Z1_EXEONLYRAM address location in OTP. 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)
3	EXEONLY_RAM3	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[3:3] when a read is issued to Z1_EXEONLYRAM address location in OTP. 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)

Table 2-210. Z1_EXEONLYRAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	EXEONLY_RAM2	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[2:2] when a read is issued to Z1_EXEONLYRAM address location in OTP. 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)
1	EXEONLY_RAM1	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[1:1] when a read is issued to Z1_EXEONLYRAM address location in OTP. 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)
0	EXEONLY_RAM0	R	0h	Value in this field gets loaded from Z1_EXEONLYRAM[0:0] when a read is issued to Z1_EXEONLYRAM address location in OTP. 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)

2.14.15 DCSM_Z2_REGS Registers

Table 2-211 lists the memory-mapped registers for the DCSM_Z2_REGS. All register offset addresses not listed in Table 2-211 should be considered as reserved locations and the register contents should not be modified.

Table 2-211. DCSM_Z2_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	Z2_LINKPOINTER	Zone 2 Link Pointer		Go
2h	Z2_OTPSECLOCK	Zone 2 OTP Secure JTAG lock		Go
4h	Z2_BOOTCTRL	Boot Mode		Go
6h	Z2_LINKPOINTERERR	Link Pointer Error		Go
10h	Z2_CSMKEY0	Zone 2 CSM Key 0		Go
12h	Z2_CSMKEY1	Zone 2 CSM Key 1		Go
14h	Z2_CSMKEY2	Zone 2 CSM Key 2		Go
16h	Z2_CSMKEY3	Zone 2 CSM Key 3		Go
19h	Z2_CR	Zone 2 CSM Control Register		Go
1Ah	Z2_GRABSECTR	Zone 2 Grab Flash Sectors Register		Go
1Ch	Z2_GRABRAMR	Zone 2 Grab RAM Blocks Register		Go
1Eh	Z2_EXEONLYSECTR	Zone 2 Flash Execute_Only Sector Register		Go
20h	Z2_EXEONLYRAMR	Zone 2 RAM Execute_Only Block Register		Go

2.14.15.1 Z2_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

Z2_LINKPOINTER is shown in [Figure 2-204](#) and described in [Table 2-212](#).

Zone 2 Link Pointer

Figure 2-204. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-212. Z2_LINKPOINTER Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R	7h	Reserved
28-0	LINKPOINTER	R	0h	This is the Resolved Link-Pointer value which is generated by looking at the three physical Link-Pointer values loaded from OTP.

2.14.15.2 Z2_OTPSECLOCK Register (Offset = 2h) [reset = FFFh]

Z2_OTPSECLOCK is shown in [Figure 2-205](#) and described in [Table 2-213](#).

Zone 2 OTP Secure JTAG lock

Figure 2-205. 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-Fh				R-Fh				R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-213. 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	Fh	Value in this field gets loaded from Z2_CRCLOCK[11:8] when a read is issued to address location of Z2_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.
7-4	PSWDLOCK	R	Fh	Value in this field gets loaded from Z2_PSWDLOCK[7:4] when a read is issued to address location of Z2_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.
3-0	RESERVED	R	0h	Reserved

2.14.15.3 Z2_BOOTCTRL Register (Offset = 4h) [reset = 0h]

Z2_BOOTCTRL is shown in [Figure 2-206](#) and described in [Table 2-214](#).

Boot Mode

Figure 2-206. Z2_BOOTCTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTPIN1								BOOTPIN0								BMODE								KEY							
R-0h								R-0h								R-0h								R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-214. Z2_BOOTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	BOOTPIN1	R	0h	<p>This field gets loaded with Z2_BOOTCTRL[31:24] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.</p> <p>This assigns the pin to be used as BOOTPIN1.</p> <p>0 : Pick default bootmode pin.</p> <p>1 : Pick GPIO0 as BOOTPIN1.</p> <p>2 : Pick GPIO1 as BOOTPIN1.</p> <p>....</p> <p>....</p> <p>n : Pick GPIO_n-1 as BOOTPIN1.</p>
23-16	BOOTPIN0	R	0h	<p>This field gets loaded with Z2_BOOTCTRL[23:16] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.</p> <p>This assigns the pin to be used as BOOTPIN1.</p> <p>0 : Pick default bootmode pin.</p> <p>1 : Pick GPIO0 as BOOTPIN1.</p> <p>2 : Pick GPIO1 as BOOTPIN1.</p> <p>....</p> <p>....</p> <p>n : Pick GPIO_n-1 as BOOTPIN1.</p>
15-8	BMODE	R	0h	<p>This field gets loaded with Z2_BOOTCTRL[16:8] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.</p>
7-0	KEY	R	0h	<p>This field gets loaded with Z2_BOOTCTRL[7:0] when a dummy read is issued to address location of Z2_BOOTCTRL in OTP.</p>

2.14.15.4 Z2_LINKPOINTERERR Register (Offset = 6h) [reset = FFFFFFFFh]

Z2_LINKPOINTERERR is shown in [Figure 2-207](#) and described in [Table 2-215](#).

Link Pointer Error

Figure 2-207. Z2_LINKPOINTERERR Register

31	30	29	28	27	26	25	24	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_LINKPOINTERERR																															
R-FFFFFFFh																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-215. Z2_LINKPOINTERERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	Z2_LINKPOINTERERR	R	FFFFFFFh	<p>These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP.</p> <p>0 : No Error.</p> <p>Other : Error on bit positions which is set to 1.</p>

2.14.15.5 Z2_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z2_CSMKEY0 is shown in [Figure 2-208](#) and described in [Table 2-216](#).

Zone 2 CSM Key 0

Figure 2-208. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-216. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.15.6 Z2_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z2_CSMKEY1 is shown in [Figure 2-209](#) and described in [Table 2-217](#).

Zone 2 CSM Key 1

Figure 2-209. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-217. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.15.7 Z2_CSMKEY2 Register (Offset = 14h) [reset = 0h]

Z2_CSMKEY2 is shown in [Figure 2-210](#) and described in [Table 2-218](#).

Zone 2 CSM Key 2

Figure 2-210. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-218. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.15.8 Z2_CSMKEY3 Register (Offset = 16h) [reset = 0h]

Z2_CSMKEY3 is shown in [Figure 2-211](#) and described in [Table 2-219](#).

Zone 2 CSM Key 3

Figure 2-211. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-219. 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 OTP (zone gets unlock only if 128 bit password in OTP match with value written in four CSMKEY registers.)

2.14.15.9 Z2_CR Register (Offset = 19h) [reset = 8h]

Z2_CR is shown in [Figure 2-212](#) and described in [Table 2-220](#).

Zone 2 CSM Control Register

Figure 2-212. 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-0h	R-1h	R-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-220. 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.
14-8	RESERVED	R=0	0h	Reserved
7	RESERVED	R	0h	Reserved
6	ARMED	R	0h	0 : Dummy read to CSM Password locations in OTP hasn't been performed. 1 : Dummy read to CSM Password locations in OTP has been performed.
5	UNSECURE	R	0h	Indicates the state of Zone. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state.
4	ALLONE	R	0h	Indicates the state of CSM passowrds. 0 : CSM Passwords are not all ones. 1 : CSM Passwords are all ones and zone is in unlock state.
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.
2-0	RESERVED	R	0h	Reserved

2.14.15.10 Z2_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

Z2_GRABSECTR is shown in [Figure 2-213](#) and described in [Table 2-221](#).

Zone 2 Grab Flash Sectors Register

Figure 2-213. Z2_GRABSECTR Register

31	30	29	28	27	26	25	24
RESERVED		GRAB_BANK2		GRAB_SECTN		GRAB_SECTM	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECTL		GRAB_SECTK		GRAB_SECTJ		GRAB_SECTI	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECTH		GRAB_SECTG		GRAB_SECTF		GRAB_SECTE	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECTD		GRAB_SECTC		GRAB_SECTB		GRAB_SECTA	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-221. Z2_GRABSECTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	GRAB_BANK2	R	0h	Value in this field gets loaded from Z2_GRABSECTR[29:28] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash BANK2 J is inaccessible. 01 : Request to allocate Flash BANK2 to Zone2. 10 : Request to allocate Flash BANK2 to Zone2. 11 : Request to make Flash sector BANK2 Non-Secure.
27-26	GRAB_SECTN	R	0h	Value in this field gets loaded from Z2_GRABSECTR[27:26] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector N is inaccessible. 01 : Request to allocate Flash Sector N to Zone2. 10 : Request to allocate Flash Sector N to Zone2. 11 : Request to make Flash sector N Non-Secure.
25-24	GRAB_SECTM	R	0h	Value in this field gets loaded from Z2_GRABSECTR[25:24] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector M is inaccessible. 01 : Request to allocate Flash Sector M to Zone2. 10 : Request to allocate Flash Sector M to Zone2. 11 : Request to make Flash sector M Non-Secure.
23-22	GRAB_SECTL	R	0h	Value in this field gets loaded from Z2_GRABSECTR[23:22] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector L is inaccessible. 01 : Request to allocate Flash Sector L to Zone2. 10 : Request to allocate Flash Sector L to Zone2. 11 : Request to make Flash sector L Non-Secure.

Table 2-221. Z2_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	GRAB_SECTK	R	0h	Value in this field gets loaded from Z2_GRABSECTR[21:20] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector K is inaccessible. 01 : Request to allocate Flash Sector K to Zone2. 10 : Request to allocate Flash Sector K to Zone2. 11 : Request to make Flash sector K Non-Secure.
19-18	GRAB_SECTJ	R	0h	Value in this field gets loaded from Z2_GRABSECTR[19:18] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector J is inaccessible. 01 : Request to allocate Flash Sector J to Zone2. 10 : Request to allocate Flash Sector J to Zone2. 11 : Request to make Flash sector J Non-Secure.
17-16	GRAB_SECTI	R	0h	Value in this field gets loaded from Z2_GRABSECTR[17:16] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector I is inaccessible. 01 : Request to allocate Flash Sector I to Zone2. 10 : Request to allocate Flash Sector I to Zone2. 11 : Request to make Flash sector I Non-Secure.
15-14	GRAB_SECTH	R	0h	Value in this field gets loaded from Z2_GRABSECTR[15:14] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector H is inaccessible. 01 : Request to allocate Flash Sector H to Zone2. 10 : Request to allocate Flash Sector H to Zone2. 11 : Request to make Flash sector H Non-Secure.
13-12	GRAB_SECTG	R	0h	Value in this field gets loaded from Z2_GRABSECTR[13:12] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector G is inaccessible. 01 : Request to allocate Flash Sector G to Zone2. 10 : Request to allocate Flash Sector G to Zone2. 11 : Request to make Flash sector G Non-Secure.
11-10	GRAB_SECTF	R	0h	Value in this field gets loaded from Z2_GRABSECTR[11:10] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector F is inaccessible. 01 : Request to allocate Flash Sector F to Zone2. 10 : Request to allocate Flash Sector F to Zone2. 11 : Request to make Flash sector F Non-Secure.
9-8	GRAB_SECTE	R	0h	Value in this field gets loaded from Z2_GRABSECTR[9:8] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector E is inaccessible. 01 : Request to allocate Flash Sector E to Zone2. 10 : Request to allocate Flash Sector E to Zone2. 11 : Request to make Flash sector E Non-Secure.
7-6	GRAB_SECTD	R	0h	Value in this field gets loaded from Z2_GRABSECTR[7:6] when a read is issued to address location of Z2_GRABSECTR in OTP. 00 : Invalid. Flash Sector D is inaccessible. 01 : Request to allocate Flash Sector D to Zone2. 10 : Request to allocate Flash Sector D to Zone2. 11 : Request to make Flash sector D Non-Secure.

Table 2-221. Z2_GRABSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	GRAB_SECTC	R	0h	Value in this field gets loaded from Z2_GRABSECT[5:4] when a read is issued to address location of Z2_GRABSECT in OTP. 00 : Invalid. Flash Sector C is inaccessible. 01 : Request to allocate Flash Sector C to Zone2. 10 : Request to allocate Flash Sector C to Zone2. 11 : Request to make Flash sector C Non-Secure.
3-2	GRAB_SECTB	R	0h	Value in this field gets loaded from Z2_GRABSECT[3:2] when a read is issued to address location of Z2_GRABSECT in OTP. 00 : Invalid. Flash Sector B is inaccessible. 01 : Request to allocate Flash Sector B to Zone2. 10 : Request to allocate Flash Sector B to Zone2. 11 : Request to make Flash sector B Non-Secure.
1-0	GRAB_SECTA	R	0h	Value in this field gets loaded from Z2_GRABSECT[1:0] when a read is issued to address location of Z2_GRABSECT in OTP. 00 : Invalid. Flash Sector A is inaccessible. 01 : Request to allocate Flash Sector A to Zone2. 10 : Request to allocate Flash Sector A to Zone2. 11 : Request to make Flash sector A Non-Secure.

2.14.15.11 Z2_GRABRAMR Register (Offset = 1Ch) [reset = 0h]

Z2_GRABRAMR is shown in [Figure 2-214](#) and described in [Table 2-222](#).

Zone 2 Grab RAM Blocks Register

Figure 2-214. Z2_GRABRAMR Register

31	30	29	28	27	26	25	24
RESERVED		GRAB_CLA1		RESERVED			
R-0h		R-0h		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-222. Z2_GRABRAMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	GRAB_CLA1	R	0h	Value in this field gets loaded from Z2_GRABRAM[29:28] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. CLA1 is inaccessible. 01 : Request to allocate CLA1 to Zone2. 10 : Request to allocate CLA1 to Zone2. 11 : Request to make CLA1 Non-Secure.
27-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from Z2_GRABRAM[15:14] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. D1 RAM is inaccessible. 01 : Request to allocate D1 RAM to Zone2. 10 : Request to allocate D1 RAM to Zone2. 11 : Request to make D1 RAM Non-Secure.
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from Z2_GRABRAM[13:12] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. D0 RAM is inaccessible. 01 : Request to allocate D0 RAM to Zone2. 10 : Request to allocate D0 RAM to Zone2. 11 : Request to make D0 RAM Non-Secure.
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from Z2_GRABRAM[11:10] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS5 RAM is inaccessible. 01 : Request to allocate LS5 RAM to Zone2. 10 : Request to allocate LS5 RAM to Zone2. 11 : Request to make LS5 RAM Non-Secure.

Table 2-222. Z2_GRABRAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	Value in this field gets loaded from Z2_GRABRAM[9:8] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS4 RAM is inaccessible. 01 : Request to allocate LS4 RAM to Zone2. 10 : Request to allocate LS4 RAM to Zone2. 11 : Request to make LS4 RAM Non-Secure.
7-6	GRAB_RAM3	R	0h	Value in this field gets loaded from Z2_GRABRAM[7:6] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS3 RAM is inaccessible. 01 : Request to allocate LS3 RAM to Zone2. 10 : Request to allocate LS3 RAM to Zone2. 11 : Request to make LS3 RAM Non-Secure.
5-4	GRAB_RAM2	R	0h	Value in this field gets loaded from Z2_GRABRAM[5:4] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS2 RAM is inaccessible. 01 : Request to allocate LS2 RAM to Zone2. 10 : Request to allocate LS2 RAM to Zone2. 11 : Request to make LS2 RAM Non-Secure.
3-2	GRAB_RAM1	R	0h	Value in this field gets loaded from Z2_GRABRAM[3:2] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS1 RAM is inaccessible. 01 : Request to allocate LS1 RAM to Zone2. 10 : Request to allocate LS1 RAM to Zone2. 11 : Request to make LS1 RAM Non-Secure.
1-0	GRAB_RAM0	R	0h	Value in this field gets loaded from Z2_GRABRAM[1:0] when a read is issued to address location of Z2_GRABRAM in OTP. 00 : Invalid. LS0 RAM is inaccessible. 01 : Request to allocate LS0 RAM to Zone2. 10 : Request to allocate LS0 RAM to Zone2. 11 : Request to make LS0 RAM Non-Secure.

2.14.15.12 Z2_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

Z2_EXEONLYSECTR is shown in [Figure 2-215](#) and described in [Table 2-223](#).

Zone 2 Flash Execute_Only Sector Register

Figure 2-215. 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
RESERVED	EXEONLY_BA NK2	EXEONLY_SE CTN	EXEONLY_SE CTM	EXEONLY_SE CTL	EXEONLY_SE CTK	EXEONLY_SE CTJ	EXEONLY_SE CTI
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 CTH	EXEONLY_SE CTG	EXEONLY_SE CTF	EXEONLY_SE CTE	EXEONLY_SE CTD	EXEONLY_SE CTC	EXEONLY_SE CTB	EXEONLY_SE CTA
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-223. Z2_EXEONLYSECTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	EXEONLY_BANK2	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[14:14] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash BANK2 (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash BANK2 (only if it's allocated to Zone2)
13	EXEONLY_SECTN	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[13:13] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector N (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector N (only if it's allocated to Zone2)
12	EXEONLY_SECTM	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[12:12] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector M (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector M (only if it's allocated to Zone2)
11	EXEONLY_SECTL	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[11:11] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector L (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector L (only if it's allocated to Zone2)
10	EXEONLY_SECTK	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[10:10] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector K (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector K (only if it's allocated to Zone2)

Table 2-223. Z2_EXEONLYSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	EXEONLY_SECTJ	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[9:9] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector J (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector J (only if it's allocated to Zone2)
8	EXEONLY_SECTI	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[8:8] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector I (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector I (only if it's allocated to Zone2)
7	EXEONLY_SECTH	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[7:7] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector H (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector H (only if it's allocated to Zone2)
6	EXEONLY_SECTG	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[6:6] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector G (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector G (only if it's allocated to Zone2)
5	EXEONLY_SECTF	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[5:5] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector F (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector F (only if it's allocated to Zone2)
4	EXEONLY_SECTE	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[4:4] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector E (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector E (only if it's allocated to Zone2)
3	EXEONLY_SECTD	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[3:3] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector D (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector D (only if it's allocated to Zone2)
2	EXEONLY_SECTC	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[2:2] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector C (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector C (only if it's allocated to Zone2)
1	EXEONLY_SECTB	R	0h	Value in this field gets loaded from Z2_EXEONLYSECTR[1:1] when a read is issued to Z2_EXEONLYSECTR address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector B (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector B (only if it's allocated to Zone2)

Table 2-223. Z2_EXEONLYSECTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	EXEONLY_SECTA	R	0h	Value in this field gets loaded from Z2_EXEONLYSECT[0:0] when a read is issued to Z2_EXEONLYSECT address location in OTP. 0 : Execute-Only protection is enabled for Flash Sector A (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for Flash Sector A (only if it's allocated to Zone2)

2.14.15.13 Z2_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]

Z2_EXEONLYRAMR is shown in [Figure 2-216](#) and described in [Table 2-224](#).

Zone 2 RAM Execute_Only Block Register

Figure 2-216. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-224. 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 Z2_EXEONLYRAM[7:7] when a read is issued to Z2_EXEONLYRAM address location in OTP. 0 : Execute-Only protection is enabled for D1 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for D1 RAM (only if it's allocated to Zone2)
6	EXEONLY_RAM6	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[6:6] when a read is issued to Z2_EXEONLYRAM address location in OTP. 0 : Execute-Only protection is enabled for D0 RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for D0 RAM (only if it's allocated to Zone2)
5	EXEONLY_RAM5	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[5:5] when a read is issued to Z2_EXEONLYRAM address location in 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)
4	EXEONLY_RAM4	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[4:4] when a read is issued to Z2_EXEONLYRAM address location in 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)
3	EXEONLY_RAM3	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[3:3] when a read is issued to Z2_EXEONLYRAM address location in 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)

Table 2-224. Z2_EXEONLYRAMR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	EXEONLY_RAM2	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[2:2] when a read is issued to Z2_EXEONLYRAM address location in 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)
1	EXEONLY_RAM1	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[1:1] when a read is issued to Z2_EXEONLYRAM address location in 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)
0	EXEONLY_RAM0	R	0h	Value in this field gets loaded from Z2_EXEONLYRAM[0:0] when a read is issued to Z2_EXEONLYRAM address location in 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)

2.14.16 DCSM_COMMON_REGS Registers

Table 2-225 lists the memory-mapped registers for the DCSM_COMMON_REGS. All register offset addresses not listed in Table 2-225 should be considered as reserved locations and the register contents should not be modified.

Table 2-225. DCSM_COMMON_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	FLSEM	Flash Wrapper Semaphore Register		Go
2h	SECTSTAT	Sectors Status Register		Go
4h	RAMSTAT	RAM Status Register		Go

2.14.16.1 FLSEM Register (Offset = 0h) [reset = 0h]

FLSEM is shown in [Figure 2-217](#) and described in [Table 2-226](#).

Flash Wrapper Semaphore Register

Figure 2-217. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-226. 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.
7-2	RESERVED	R	0h	Reserved
1-0	SEM	R/W	0h	<p>00 : C28X Flash Wrapper registers can be written by code running from anywhere without any restriction.</p> <p>01 : Flash Wrapper registers can be written by code running from Zone1 security zone.</p> <p>10 : C28X Flash Wrapper registers can be written by code running from Zone2 security zone</p> <p>11 : C28X Flash Wrapper registers can be written by code running from anywhere without any restriction</p> <p>Allowed State Transitions in this field.</p> <p>00 TO 11 : Code running from anywhere.</p> <p>11 TO 00 : Not allowed.</p> <p>00/11 TO 01 : Code running from Zone1 only can perform this transition.</p> <p>01 TO 00/11 : Code running from Zone1 only can perform this transition.</p> <p>00/11 TO 10 : Code running from Zone2 only can perform this transition.</p> <p>10 TO 00/11 : Code running from Zone2 can perform this transition</p>

2.14.16.2 SECTSTAT Register (Offset = 2h) [reset = 0h]

SECTSTAT is shown in [Figure 2-218](#) and described in [Table 2-227](#).

Sectors Status Register

Figure 2-218. SECTSTAT Register

31	30	29	28	27	26	25	24
RESERVED		STATUS_BANK2		STATUS_SECTN		STATUS_SECTM	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECTL		STATUS_SECTK		STATUS_SECTJ		STATUS_SECTI	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECTH		STATUS_SECTG		STATUS_SECTF		STATUS_SECTE	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_SECTD		STATUS_SECTC		STATUS_SECTB		STATUS_SECTA	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-227. SECTSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	STATUS_BANK2	R	0h	Reflects the status of flash BANK2. 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.
27-26	STATUS_SECTN	R	0h	Reflects the status of flash sector N. 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.
25-24	STATUS_SECTM	R	0h	Reflects the status of flash sector M. 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.
23-22	STATUS_SECTL	R	0h	Reflects the status of flash sector L. 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.

Table 2-227. SECTSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	STATUS_SECTK	R	0h	Reflects the status of flash sector K. 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.
19-18	STATUS_SECTJ	R	0h	Reflects the status of flash sector J. 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.
17-16	STATUS_SECTI	R	0h	Reflects the status of flash sector I. 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.
15-14	STATUS_SECTH	R	0h	Reflects the status of flash sector H. 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.
13-12	STATUS_SECTG	R	0h	Reflects the status of flash sector G. 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.
11-10	STATUS_SECTF	R	0h	Reflects the status of flash sector F. 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.
9-8	STATUS_SECTE	R	0h	Reflects the status of flash sector E. 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.
7-6	STATUS_SECTD	R	0h	Reflects the status of flash sector D. 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.

Table 2-227. SECTSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	STATUS_SECTC	R	0h	Reflects the status of flash sector C. 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.
3-2	STATUS_SECTB	R	0h	Reflects the status of flash sector B. 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.
1-0	STATUS_SECTA	R	0h	Reflects the status of flash sector A. 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.

2.14.16.3 RAMSTAT Register (Offset = 4h) [reset = 0h]

RAMSTAT is shown in [Figure 2-219](#) and described in [Table 2-228](#).

RAM Status Register

Figure 2-219. RAMSTAT Register

31	30	29	28	27	26	25	24
RESERVED		STATUS_CLA1		RESERVED			
R-0h		R-0h		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-228. RAMSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	STATUS_CLA1	R	0h	Reflects the status of CLA1. 00 : CLA is in-accessible 01 : CLA belongs to Zone1. 10 : CLA belongs to Zone2. 11: CLA is un-secure and code running in both zone have full access to it.
27-16	RESERVED	R	0h	Reserved
15-14	STATUS_RAM7	R	0h	Reflects the status of D1 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.
13-12	STATUS_RAM6	R	0h	Reflects the status of D0 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.
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.

Table 2-228. RAMSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.
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.
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.
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.

2.14.17 MEM_CFG_REGS Registers

Table 2-229 lists the memory-mapped registers for the MEM_CFG_REGS. All register offset addresses not listed in Table 2-229 should be considered as reserved locations and the register contents should not be modified.

Table 2-229. MEM_CFG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	DxLOCK	Dedicated RAM Config Lock Register	EALLOW	Go
2h	DxCOMMIT	Dedicated RAM Config Lock Commit Register	EALLOW	Go
8h	DxACCPROT0	Dedicated RAM Config Register	EALLOW	Go
10h	DxTEST	Dedicated RAM TEST Register	EALLOW	Go
12h	DxINIT	Dedicated RAM Init Register	EALLOW	Go
14h	DxINITDONE	Dedicated RAM InitDone Status Register		Go
20h	LSxLOCK	Local Shared RAM Config Lock Register	EALLOW	Go
22h	LSxCOMMIT	Local Shared RAM Config Lock Commit Register	EALLOW	Go
24h	LSxMSEL	Local Shared RAM Master Sel Register	EALLOW	Go
26h	LSxCLAPGM	Local Shared RAM Prog/Exe control Register	EALLOW	Go
28h	LSxACCPROT0	Local Shared RAM Config Register 0	EALLOW	Go
2Ah	LSxACCPROT1	Local Shared RAM Config Register 1	EALLOW	Go
30h	LSxTEST	Local Shared RAM TEST Register	EALLOW	Go
32h	LSxINIT	Local Shared RAM Init Register	EALLOW	Go
34h	LSxINITDONE	Local Shared RAM InitDone Status Register		Go
40h	GSxLOCK	Global Shared RAM Config Lock Register	EALLOW	Go
42h	GSxCOMMIT	Global Shared RAM Config Lock Commit Register	EALLOW	Go
44h	GSxMSEL	Global Shared RAM Master Sel Register	EALLOW	Go
48h	GSxACCPROT0	Global Shared RAM Config Register 0	EALLOW	Go
4Ah	GSxACCPROT1	Global Shared RAM Config Register 1	EALLOW	Go
4Ch	GSxACCPROT2	Global Shared RAM Config Register 2	EALLOW	Go
4Eh	GSxACCPROT3	Global Shared RAM Config Register 3	EALLOW	Go
50h	GSxTEST	Global Shared RAM TEST Register	EALLOW	Go
52h	GSxINIT	Global Shared RAM Init Register	EALLOW	Go
54h	GSxINITDONE	Global Shared RAM InitDone Status Register		Go
70h	MSGxTEST	Message RAM TEST Register	EALLOW	Go
72h	MSGxINIT	Message RAM Init Register	EALLOW	Go
74h	MSGxINITDONE	Message RAM InitDone Status Register		Go

2.14.17.1 DxLOCK Register (Offset = 0h) [reset = 0h]

DxLOCK is shown in [Figure 2-220](#) and described in [Table 2-230](#).

Dedicated RAM Config Lock Register

Figure 2-220. 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				LOCK_D1	LOCK_D0	RESERVED	
R-0h				R/W-0h	R/W-0h	R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-230. DxLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	LOCK_D1	R/W	0h	Locks the write to access protection and master select fields for D1 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
2	LOCK_D0	R/W	0h	Locks the write to access protection and master select fields for D0 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
1-0	RESERVED	R	0h	Reserved

2.14.17.2 DxCOMMIT Register (Offset = 2h) [reset = 0h]

DxCOMMIT is shown in [Figure 2-221](#) and described in [Table 2-231](#).

Dedicated RAM Config Lock Commit Register

Figure 2-221. 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				COMMIT_D1	COMMIT_D0	RESERVED	
R-0h				R/SOnce-0h	R/SOnce-0h	R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-231. DxCOMMIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	COMMIT_D1	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for D1 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
2	COMMIT_D0	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for D0 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
1-0	RESERVED	R	0h	Reserved

2.14.17.3 DxACCPROT0 Register (Offset = 8h) [reset = 0h]

DxACCPROT0 is shown in [Figure 2-222](#) and described in [Table 2-232](#).

Dedicated RAM Config Register

Figure 2-222. DxACCPROT0 Register

31	30	29	28	27	26	25	24
RESERVED						CPUWRPROT_D1	FETCHPROT_D1
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						CPUWRPROT_D0	FETCHPROT_D0
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-232. DxACCPROT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	CPUWRPROT_D1	R/W	0h	CPU WR Protection For D1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
24	FETCHPROT_D1	R/W	0h	Fetch Protection For D1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
23-18	RESERVED	R	0h	Reserved
17	CPUWRPROT_D0	R/W	0h	CPU WR Protection For D0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are block.
16	FETCHPROT_D0	R/W	0h	Fetch Protection For D0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
15-0	RESERVED	R	0h	Reserved

2.14.17.4 DxTEST Register (Offset = 10h) [reset = 0h]

DxTEST is shown in [Figure 2-223](#) and described in [Table 2-233](#).

Dedicated RAM TEST Register

Figure 2-223. 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
TEST_D1		TEST_D0		TEST_M1		TEST_M0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-233. DxTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7-6	TEST_D1	R/W	0h	Selects the defferent modes for D1 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.
5-4	TEST_D0	R/W	0h	Selects the defferent modes for D0 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.
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.
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.

2.14.17.5 DxINIT Register (Offset = 12h) [reset = 0h]

DxINIT is shown in [Figure 2-224](#) and described in [Table 2-234](#).

Dedicated RAM Init Register

Figure 2-224. 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				INIT_D1	INIT_D0	INIT_M1	INIT_M0
R-0h				R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-234. DxINIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	INIT_D1	R=0/W=1	0h	RAM Initialization control for D1 RAM: 0: None. 1: Start RAM Initialization.
2	INIT_D0	R=0/W=1	0h	RAM Initialization control for D0 RAM: 0: None. 1: Start RAM Initialization.
1	INIT_M1	R=0/W=1	0h	RAM Initialization control for M1 RAM: 0: None. 1: Start RAM Initialization.
0	INIT_M0	R=0/W=1	0h	RAM Initialization control for M0 RAM: 0: None. 1: Start RAM Initialization.

2.14.17.6 DxINITDONE Register (Offset = 14h) [reset = 0h]

DxINITDONE is shown in [Figure 2-225](#) and described in [Table 2-235](#).

Dedicated RAM InitDone Status Register

Figure 2-225. 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				INITDONE_D1	INITDONE_D0	INITDONE_M1	INITDONE_M0
R-0h				R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-235. DxINITDONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	INITDONE_D1	R	0h	RAM Initialization status for D1 RAM: 0: RAM Initialization has completed. 1: RAM Initialization has completed.
2	INITDONE_D0	R	0h	RAM Initialization status for D0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
1	INITDONE_M1	R	0h	RAM Initialization status for M1 RAM: 0: RAM Initialization has completed. 1: RAM Initialization has completed.
0	INITDONE_M0	R	0h	RAM Initialization status for M0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.

2.14.17.7 LSxLOCK Register (Offset = 20h) [reset = 0h]

LSxLOCK is shown in [Figure 2-226](#) and described in [Table 2-236](#).

Local Shared RAM Config Lock Register

Figure 2-226. 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
RESERVED		LOCK_LS5	LOCK_LS4	LOCK_LS3	LOCK_LS2	LOCK_LS1	LOCK_LS0
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-236. LSxLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	LOCK_LS5	R/W	0h	Locks the write to access protection and master select fields for LS5 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
4	LOCK_LS4	R/W	0h	Locks the write to access protection and master select fields for LS4 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
3	LOCK_LS3	R/W	0h	Locks the write to access protection and master select fields for LS3 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
2	LOCK_LS2	R/W	0h	Locks the write to access protection and master select fields for LS2 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
1	LOCK_LS1	R/W	0h	Locks the write to access protection and master select fields for LS1 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
0	LOCK_LS0	R/W	0h	Locks the write to access protection and master select fields for LS0 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.

2.14.17.8 LSxCOMMIT Register (Offset = 22h) [reset = 0h]

LSxCOMMIT is shown in [Figure 2-227](#) and described in [Table 2-237](#).

Local Shared RAM Config Lock Commit Register

Figure 2-227. 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
RESERVED		COMMIT_LS5	COMMIT_LS4	COMMIT_LS3	COMMIT_LS2	COMMIT_LS1	COMMIT_LS0
R-0h		R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-237. LSxCOMMIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	COMMIT_LS5	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS5 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
4	COMMIT_LS4	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS4 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
3	COMMIT_LS3	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS3 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
2	COMMIT_LS2	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS2 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
1	COMMIT_LS1	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS1 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.

Table 2-237. LSxCOMMIT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	COMMIT_LS0	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for LS0 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.

2.14.17.9 LSxMSEL Register (Offset = 24h) [reset = 0h]

LSxMSEL is shown in [Figure 2-228](#) and described in [Table 2-238](#).

Local Shared RAM Master Sel Register

Figure 2-228. 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
RESERVED				MSEL_LS5		MSEL_LS4	
R-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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-238. LSxMSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
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. 10: Reserved. 11: Reserved.
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. 10: Reserved. 11: Reserved.
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. 10: Reserved. 11: Reserved.
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. 10: Reserved. 11: Reserved.
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. 10: Reserved. 11: Reserved.

Table 2-238. LSxMSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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. 10: Reserved. 11: Reserved.

2.14.17.10 LSxCLAPGM Register (Offset = 26h) [reset = 0h]

LSxCLAPGM is shown in [Figure 2-229](#) and described in [Table 2-239](#).

Local Shared RAM Prog/Exe control Register

Figure 2-229. 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
RESERVED		CLAPGM_LS5	CLAPGM_LS4	CLAPGM_LS3	CLAPGM_LS2	CLAPGM_LS1	CLAPGM_LS0
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-239. LSxCLAPGM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	CLAPGM_LS5	R/W	0h	Selects LS5 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.
4	CLAPGM_LS4	R/W	0h	Selects LS4 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.
3	CLAPGM_LS3	R/W	0h	Selects LS3 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.
2	CLAPGM_LS2	R/W	0h	Selects LS2 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.
1	CLAPGM_LS1	R/W	0h	Selects LS1 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.
0	CLAPGM_LS0	R/W	0h	Selects LS0 RAM as program vs data memory for CLA: 0: CLA Data memory. 1: CLA Program memory.

2.14.17.11 LSxACCPROT0 Register (Offset = 28h) [reset = 0h]

LSxACCPROT0 is shown in [Figure 2-230](#) and described in [Table 2-240](#).

Local Shared RAM Config Register 0

Figure 2-230. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-240. 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.
24	FETCHPROT_LS3	R/W	0h	Fetch Protection For LS3 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.
16	FETCHPROT_LS2	R/W	0h	Fetch Protection For LS2 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.
8	FETCHPROT_LS1	R/W	0h	Fetch Protection For LS1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
7-2	RESERVED	R	0h	Reserved

Table 2-240. LSxACCPROT0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	CPUWRPROT_LS0	R/W	0h	CPU WR Protection For LS0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_LS0	R/W	0h	Fetch Protection For LS0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.12 LSxACCPROT1 Register (Offset = 2Ah) [reset = 0h]

LSxACCPROT1 is shown in [Figure 2-231](#) and described in [Table 2-241](#).

Local Shared RAM Config Register 1

Figure 2-231. LSxACCPROT1 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						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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-241. LSxACCPROT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
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.
8	FETCHPROT_LS5	R/W	0h	Fetch Protection For LS5 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.
0	FETCHPROT_LS4	R/W	0h	Fetch Protection For LS4 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.13 LSxTEST Register (Offset = 30h) [reset = 0h]

LSxTEST is shown in [Figure 2-232](#) and described in [Table 2-242](#).

Local Shared RAM TEST Register

Figure 2-232. 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
RESERVED				TEST_LS5		TEST_LS4	
R-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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-242. LSxTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
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.
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.
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.
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.
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.

Table 2-242. LSxTEST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	TEST_LS0	R/W	0h	<p>Selects the different modes for LS0 RAM:</p> <p>00: Functional Mode.</p> <p>01: Writes are allowed to data bits only. No write to parity bits.</p> <p>10: Writes are allowed to parity bits only. No write to data bits.</p> <p>11: Functional Mode.</p>

2.14.17.14 LSxINIT Register (Offset = 32h) [reset = 0h]

LSxINIT is shown in [Figure 2-233](#) and described in [Table 2-243](#).

Local Shared RAM Init Register

Figure 2-233. 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
RESERVED		INIT_LS5	INIT_LS4	INIT_LS3	INIT_LS2	INIT_LS1	INIT_LS0
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-243. LSxINIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	INIT_LS5	R=0/W=1	0h	RAM Initialization control for LS5 RAM: 0: None. 1: Start RAM Initialization.
4	INIT_LS4	R=0/W=1	0h	RAM Initialization control for LS4 RAM: 0: None. 1: Start RAM Initialization.
3	INIT_LS3	R=0/W=1	0h	RAM Initialization control for LS3 RAM: 0: None. 1: Start RAM Initialization.
2	INIT_LS2	R=0/W=1	0h	RAM Initialization control for LS2 RAM: 0: None. 1: Start RAM Initialization.
1	INIT_LS1	R=0/W=1	0h	RAM Initialization control for LS1 RAM: 0: None. 1: Start RAM Initialization.
0	INIT_LS0	R=0/W=1	0h	RAM Initialization control for LS0 RAM: 0: None. 1: Start RAM Initialization.

2.14.17.15 LSxINITDONE Register (Offset = 34h) [reset = 0h]

LSxINITDONE is shown in [Figure 2-234](#) and described in [Table 2-244](#).

Local Shared RAM InitDone Status Register

Figure 2-234. 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
RESERVED		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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-244. LSxINITDONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	INITDONE_LS5	R	0h	RAM Initialization status for LS5 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
4	INITDONE_LS4	R	0h	RAM Initialization status for LS4 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
3	INITDONE_LS3	R	0h	RAM Initialization status for LS3 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
2	INITDONE_LS2	R	0h	RAM Initialization status for LS2 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
1	INITDONE_LS1	R	0h	RAM Initialization status for LS1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
0	INITDONE_LS0	R	0h	RAM Initialization status for LS0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.

2.14.17.16 GSxLOCK Register (Offset = 40h) [reset = 0h]

GSxLOCK is shown in [Figure 2-235](#) and described in [Table 2-245](#).

Global Shared RAM Config Lock Register

Figure 2-235. 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
LOCK_GS15	LOCK_GS14	LOCK_GS13	LOCK_GS12	LOCK_GS11	LOCK_GS10	LOCK_GS9	LOCK_GS8
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
LOCK_GS7	LOCK_GS6	LOCK_GS5	LOCK_GS4	LOCK_GS3	LOCK_GS2	LOCK_GS1	LOCK_GS0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-245. GSxLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	LOCK_GS15	R/W	0h	Locks the write to access protection and master select fields for GS15 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
14	LOCK_GS14	R/W	0h	Locks the write to access protection and master select fields for GS14 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
13	LOCK_GS13	R/W	0h	Locks the write to access protection and master select fields for GS13 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
12	LOCK_GS12	R/W	0h	Locks the write to access protection and master select fields for GS12 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
11	LOCK_GS11	R/W	0h	Locks the write to access protection and master select fields for GS11 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
10	LOCK_GS10	R/W	0h	Locks the write to access protection and master select fields for GS10 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
9	LOCK_GS9	R/W	0h	Locks the write to access protection and master select fields for GS9 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.

Table 2-245. GSxLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	LOCK_GS8	R/W	0h	Locks the write to access protection and master select fields for GS8 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
7	LOCK_GS7	R/W	0h	Locks the write to access protection and master select fields for GS7 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
6	LOCK_GS6	R/W	0h	Locks the write to access protection and master select fields for GS6 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
5	LOCK_GS5	R/W	0h	Locks the write to access protection and master select fields for GS5 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
4	LOCK_GS4	R/W	0h	Locks the write to access protection and master select fields for GS4 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
3	LOCK_GS3	R/W	0h	Locks the write to access protection and master select fields for GS3 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
2	LOCK_GS2	R/W	0h	Locks the write to access protection and master select fields for GS2 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
1	LOCK_GS1	R/W	0h	Locks the write to access protection and master select fields for GS1 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.
0	LOCK_GS0	R/W	0h	Locks the write to access protection and master select fields for GS0 RAM: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.

2.14.17.17 GSxCOMMIT Register (Offset = 42h) [reset = 0h]

GSxCOMMIT is shown in [Figure 2-236](#) and described in [Table 2-246](#).

Global Shared RAM Config Lock Commit Register

Figure 2-236. 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
COMMIT_GS1 5	COMMIT_GS1 4	COMMIT_GS1 3	COMMIT_GS1 2	COMMIT_GS1 1	COMMIT_GS1 0	COMMIT_GS9	COMMIT_GS8
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h
7	6	5	4	3	2	1	0
COMMIT_GS7	COMMIT_GS6	COMMIT_GS5	COMMIT_GS4	COMMIT_GS3	COMMIT_GS2	COMMIT_GS1	COMMIT_GS0
R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-246. GSxCOMMIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	COMMIT_GS15	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS15 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
14	COMMIT_GS14	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS14 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
13	COMMIT_GS13	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS13 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
12	COMMIT_GS12	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS12 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
11	COMMIT_GS11	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS11 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
10	COMMIT_GS10	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for GS10 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.

Table 2-246. GSxCOMMIT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	COMMIT_GS9	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS9 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
8	COMMIT_GS8	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS8 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
7	COMMIT_GS7	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS7 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
6	COMMIT_GS6	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS6 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
5	COMMIT_GS5	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS5 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
4	COMMIT_GS4	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS4 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
3	COMMIT_GS3	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS3 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
2	COMMIT_GS2	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS2 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
1	COMMIT_GS1	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS1 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.
0	COMMIT_GS0	R/SONce	0h	Permanently Locks the write to access protection and master select fields for GS0 RAM: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in GSxLOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.

2.14.17.18 GSxMSEL Register (Offset = 44h) [reset = 0h]

GSxMSEL is shown in [Figure 2-237](#) and described in [Table 2-247](#).

Global Shared RAM Master Sel Register

Figure 2-237. GSxMSEL 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_GS15	MSEL_GS14	MSEL_GS13	MSEL_GS12	MSEL_GS11	MSEL_GS10	MSEL_GS9	MSEL_GS8
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
MSEL_GS7	MSEL_GS6	MSEL_GS5	MSEL_GS4	MSEL_GS3	MSEL_GS2	MSEL_GS1	MSEL_GS0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-247. GSxMSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	MSEL_GS15	R/W	0h	Master Select for GS15 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
14	MSEL_GS14	R/W	0h	Master Select for GS14 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
13	MSEL_GS13	R/W	0h	Master Select for GS13 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
12	MSEL_GS12	R/W	0h	Master Select for GS12 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
11	MSEL_GS11	R/W	0h	Master Select for GS11 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
10	MSEL_GS10	R/W	0h	Master Select for GS10 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
9	MSEL_GS9	R/W	0h	Master Select for GS9 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
8	MSEL_GS8	R/W	0h	Master Select for GS8 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
7	MSEL_GS7	R/W	0h	Master Select for GS7 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.

Table 2-247. GSxMSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	MSEL_GS6	R/W	0h	Master Select for GS6 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
5	MSEL_GS5	R/W	0h	Master Select for GS5 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
4	MSEL_GS4	R/W	0h	Master Select for GS4 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
3	MSEL_GS3	R/W	0h	Master Select for GS3 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
2	MSEL_GS2	R/W	0h	Master Select for GS2 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
1	MSEL_GS1	R/W	0h	Master Select for GS1 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.
0	MSEL_GS0	R/W	0h	Master Select for GS0 RAM: 0: CPU1 is master for this memory. 1: CPU2 is master for this memory.

2.14.17.19 GSxACCPROT0 Register (Offset = 48h) [reset = 0h]

GSxACCPROT0 is shown in [Figure 2-238](#) and described in [Table 2-248](#).

Global Shared RAM Config Register 0

Figure 2-238. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-248. 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.
25	CPUWRPROT_GS3	R/W	0h	CPU WR Protection For GS3 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
24	FETCHPROT_GS3	R/W	0h	Fetch Protection For GS3 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.
17	CPUWRPROT_GS2	R/W	0h	CPU WR Protection For GS2 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
16	FETCHPROT_GS2	R/W	0h	Fetch Protection For GS2 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.

Table 2-248. GSxACCPROT0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	CPUWRPROT_GS1	R/W	0h	CPU WR Protection For GS1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
8	FETCHPROT_GS1	R/W	0h	Fetch Protection For GS1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
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.
1	CPUWRPROT_GS0	R/W	0h	CPU WR Protection For GS0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_GS0	R/W	0h	Fetch Protection For GS0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.20 GSxACCPROT1 Register (Offset = 4Ah) [reset = 0h]

GSxACCPROT1 is shown in [Figure 2-239](#) and described in [Table 2-249](#).

Global Shared RAM Config Register 1

Figure 2-239. GSxACCPROT1 Register

31	30	29	28	27	26	25	24
RESERVED					DMAWRPROT_GS7	CPUWRPROT_GS7	FETCHPROT_GS7
R-0h					R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED					DMAWRPROT_GS6	CPUWRPROT_GS6	FETCHPROT_GS6
R-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED					DMAWRPROT_GS5	CPUWRPROT_GS5	FETCHPROT_GS5
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					DMAWRPROT_GS4	CPUWRPROT_GS4	FETCHPROT_GS4
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-249. GSxACCPROT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	DMAWRPROT_GS7	R/W	0h	DMA WR Protection For GS7 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
25	CPUWRPROT_GS7	R/W	0h	CPU WR Protection For GS7 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
24	FETCHPROT_GS7	R/W	0h	Fetch Protection For GS7 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
23-19	RESERVED	R	0h	Reserved
18	DMAWRPROT_GS6	R/W	0h	DMA WR Protection For GS6 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
17	CPUWRPROT_GS6	R/W	0h	CPU WR Protection For GS6 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
16	FETCHPROT_GS6	R/W	0h	Fetch Protection For GS6 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
15-11	RESERVED	R	0h	Reserved
10	DMAWRPROT_GS5	R/W	0h	DMA WR Protection For GS5 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.

Table 2-249. GSxACCPROT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	CPUWRPROT_GS5	R/W	0h	CPU WR Protection For GS5 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
8	FETCHPROT_GS5	R/W	0h	Fetch Protection For GS5 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
7-3	RESERVED	R	0h	Reserved
2	DMAWRPROT_GS4	R/W	0h	DMA WR Protection For GS4 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
1	CPUWRPROT_GS4	R/W	0h	CPU WR Protection For GS4 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_GS4	R/W	0h	Fetch Protection For GS4 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.21 GSxACCPROT2 Register (Offset = 4Ch) [reset = 0h]

GSxACCPROT2 is shown in [Figure 2-240](#) and described in [Table 2-250](#).

Global Shared RAM Config Register 2

Figure 2-240. GSxACCPROT2 Register

31	30	29	28	27	26	25	24
RESERVED					DMAWRPROT_ _GS11	CPUWRPROT_ _GS11	FETCHPROT_ _GS11
R-0h					R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED					DMAWRPROT_ _GS10	CPUWRPROT_ _GS10	FETCHPROT_ _GS10
R-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED					DMAWRPROT_ _GS9	CPUWRPROT_ _GS9	FETCHPROT_ _GS9
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					DMAWRPROT_ _GS8	CPUWRPROT_ _GS8	FETCHPROT_ _GS8
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-250. GSxACCPROT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	DMAWRPROT_GS11	R/W	0h	DMA WR Protection For GS11 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
25	CPUWRPROT_GS11	R/W	0h	CPU WR Protection For GS11 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
24	FETCHPROT_GS11	R/W	0h	Fetch Protection For GS11 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
23-19	RESERVED	R	0h	Reserved
18	DMAWRPROT_GS10	R/W	0h	DMA WR Protection For GS10 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
17	CPUWRPROT_GS10	R/W	0h	CPU WR Protection For GS10 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
16	FETCHPROT_GS10	R/W	0h	Fetch Protection For GS10 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
15-11	RESERVED	R	0h	Reserved
10	DMAWRPROT_GS9	R/W	0h	DMA WR Protection For GS9 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.

Table 2-250. GSxACCPROT2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	CPUWRPROT_GS9	R/W	0h	CPU WR Protection For GS9 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
8	FETCHPROT_GS9	R/W	0h	Fetch Protection For GS9 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
7-3	RESERVED	R	0h	Reserved
2	DMAWRPROT_GS8	R/W	0h	DMA WR Protection For GS8 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
1	CPUWRPROT_GS8	R/W	0h	CPU WR Protection For GS8 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_GS8	R/W	0h	Fetch Protection For GS8 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.22 GSxACCPROT3 Register (Offset = 4Eh) [reset = 0h]

GSxACCPROT3 is shown in [Figure 2-241](#) and described in [Table 2-251](#).

Global Shared RAM Config Register 3

Figure 2-241. GSxACCPROT3 Register

31	30	29	28	27	26	25	24
RESERVED					DMAWRPROT_GS15	CPUWRPROT_GS15	FETCHPROT_GS15
R-0h					R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED					DMAWRPROT_GS14	CPUWRPROT_GS14	FETCHPROT_GS14
R-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED					DMAWRPROT_GS13	CPUWRPROT_GS13	FETCHPROT_GS13
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					DMAWRPROT_GS12	CPUWRPROT_GS12	FETCHPROT_GS12
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-251. GSxACCPROT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26	DMAWRPROT_GS15	R/W	0h	DMA WR Protection For GS15 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
25	CPUWRPROT_GS15	R/W	0h	CPU WR Protection For GS15 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
24	FETCHPROT_GS15	R/W	0h	Fetch Protection For GS15 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
23-19	RESERVED	R	0h	Reserved
18	DMAWRPROT_GS14	R/W	0h	DMA WR Protection For GS14 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
17	CPUWRPROT_GS14	R/W	0h	CPU WR Protection For GS14 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
16	FETCHPROT_GS14	R/W	0h	Fetch Protection For GS14 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
15-11	RESERVED	R	0h	Reserved
10	DMAWRPROT_GS13	R/W	0h	DMA WR Protection For GS13 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.

Table 2-251. GSxACCPROT3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	CPUWRPROT_GS13	R/W	0h	CPU WR Protection For GS13 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
8	FETCHPROT_GS13	R/W	0h	Fetch Protection For GS13 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.
7-3	RESERVED	R	0h	Reserved
2	DMAWRPROT_GS12	R/W	0h	DMA WR Protection For GS12 RAM: 0: DMA Writes are allowed. 1: DMA Writes are blocked.
1	CPUWRPROT_GS12	R/W	0h	CPU WR Protection For GS12 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_GS12	R/W	0h	Fetch Protection For GS12 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked.

2.14.17.23 GSxTEST Register (Offset = 50h) [reset = 0h]

GSxTEST is shown in [Figure 2-242](#) and described in [Table 2-252](#).

Global Shared RAM TEST Register

Figure 2-242. GSxTEST Register

31	30	29	28	27	26	25	24
TEST_GS15		TEST_GS14		TEST_GS13		TEST_GS12	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
TEST_GS11		TEST_GS10		TEST_GS9		TEST_GS8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
TEST_GS7		TEST_GS6		TEST_GS5		TEST_GS4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-252. GSxTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	TEST_GS15	R/W	0h	Selects the different modes for GS15 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.
29-28	TEST_GS14	R/W	0h	Selects the different modes for GS14 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.
27-26	TEST_GS13	R/W	0h	Selects the different modes for GS13 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.
25-24	TEST_GS12	R/W	0h	Selects the different modes for GS12 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.
23-22	TEST_GS11	R/W	0h	Selects the different modes for GS11 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.

Table 2-252. GSxTEST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	TEST_GS10	R/W	0h	Selects the different modes for GS10 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.
19-18	TEST_GS9	R/W	0h	Selects the different modes for GS9 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.
17-16	TEST_GS8	R/W	0h	Selects the different modes for GS8 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.
15-14	TEST_GS7	R/W	0h	Selects the different modes for GS7 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.
13-12	TEST_GS6	R/W	0h	Selects the different modes for GS6 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.
11-10	TEST_GS5	R/W	0h	Selects the different modes for GS5 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.
9-8	TEST_GS4	R/W	0h	Selects the different modes for GS4 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.
7-6	TEST_GS3	R/W	0h	Selects the different 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.
5-4	TEST_GS2	R/W	0h	Selects the different 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.

Table 2-252. GSxTEST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-2	TEST_GS1	R/W	0h	<p>Selects the different modes for GS1 RAM:</p> <p>00: Functional Mode.</p> <p>01: Writes are allowed to data bits only. No write to parity bits.</p> <p>10: Writes are allowed to parity bits only. No write to data bits.</p> <p>11: Functional Mode.</p>
1-0	TEST_GS0	R/W	0h	<p>Selects the different modes for GS0 RAM:</p> <p>00: Functional Mode.</p> <p>01: Writes are allowed to data bits only. No write to parity bits.</p> <p>10: Writes are allowed to parity bits only. No write to data bits.</p> <p>11: Functional Mode.</p>

2.14.17.24 GSxINIT Register (Offset = 52h) [reset = 0h]

GSxINIT is shown in [Figure 2-243](#) and described in [Table 2-253](#).

Global Shared RAM Init Register

Figure 2-243. 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
INIT_GS15	INIT_GS14	INIT_GS13	INIT_GS12	INIT_GS11	INIT_GS10	INIT_GS9	INIT_GS8
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
INIT_GS7	INIT_GS6	INIT_GS5	INIT_GS4	INIT_GS3	INIT_GS2	INIT_GS1	INIT_GS0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-253. GSxINIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	INIT_GS15	R=0/W=1	0h	RAM Initialization control for GS15 RAM: 0: None. 1: Start RAM Initialization.
14	INIT_GS14	R=0/W=1	0h	RAM Initialization control for GS14 RAM: 0: None. 1: Start RAM Initialization.
13	INIT_GS13	R=0/W=1	0h	RAM Initialization control for GS13 RAM: 0: None. 1: Start RAM Initialization.
12	INIT_GS12	R=0/W=1	0h	RAM Initialization control for GS12 RAM: 0: None. 1: Start RAM Initialization.
11	INIT_GS11	R=0/W=1	0h	RAM Initialization control for GS11 RAM: 0: None. 1: Start RAM Initialization.
10	INIT_GS10	R=0/W=1	0h	RAM Initialization control for GS10 RAM: 0: None. 1: Start RAM Initialization.
9	INIT_GS9	R=0/W=1	0h	RAM Initialization control for GS9 RAM: 0: None. 1: Start RAM Initialization.
8	INIT_GS8	R=0/W=1	0h	RAM Initialization control for GS8 RAM: 0: None. 1: Start RAM Initialization.
7	INIT_GS7	R=0/W=1	0h	RAM Initialization control for GS7 RAM: 0: None. 1: Start RAM Initialization.

Table 2-253. GSxINIT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	INIT_GS6	R=0/W=1	0h	RAM Initialization control for GS6 RAM: 0: None. 1: Start RAM Initialization.
5	INIT_GS5	R=0/W=1	0h	RAM Initialization control for GS5 RAM: 0: None. 1: Start RAM Initialization.
4	INIT_GS4	R=0/W=1	0h	RAM Initialization control for GS4 RAM: 0: None. 1: Start RAM Initialization.
3	INIT_GS3	R=0/W=1	0h	RAM Initialization control for GS3 RAM: 0: None. 1: Start RAM Initialization.
2	INIT_GS2	R=0/W=1	0h	RAM Initialization control for GS2 RAM: 0: None. 1: Start RAM Initialization.
1	INIT_GS1	R=0/W=1	0h	RAM Initialization control for GS1 RAM: 0: None. 1: Start RAM Initialization.
0	INIT_GS0	R=0/W=1	0h	RAM Initialization control for GS0 RAM: 0: None. 1: Start RAM Initialization.

2.14.17.25 GSxINITDONE Register (Offset = 54h) [reset = 0h]

GSxINITDONE is shown in [Figure 2-244](#) and described in [Table 2-254](#).

Global Shared RAM InitDone Status Register

Figure 2-244. 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
INITDONE_GS 15	INITDONE_GS 14	INITDONE_GS 13	INITDONE_GS 12	INITDONE_GS 11	INITDONE_GS 10	INITDONE_GS 9	INITDONE_GS 8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
INITDONE_GS 7	INITDONE_GS 6	INITDONE_GS 5	INITDONE_GS 4	INITDONE_GS 3	INITDONE_GS 2	INITDONE_GS 1	INITDONE_GS 0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-254. GSxINITDONE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	INITDONE_GS15	R	0h	RAM Initialization status for GS15 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
14	INITDONE_GS14	R	0h	RAM Initialization status for GS14 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
13	INITDONE_GS13	R	0h	RAM Initialization status for GS13 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
12	INITDONE_GS12	R	0h	RAM Initialization status for GS12 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
11	INITDONE_GS11	R	0h	RAM Initialization status for GS11 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
10	INITDONE_GS10	R	0h	RAM Initialization status for GS10 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
9	INITDONE_GS9	R	0h	RAM Initialization status for GS9 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
8	INITDONE_GS8	R	0h	RAM Initialization status for GS8 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.

Table 2-254. GSxINITDONE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	INITDONE_GS7	R	0h	RAM Initialization status for GS7 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
6	INITDONE_GS6	R	0h	RAM Initialization status for GS6 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
5	INITDONE_GS5	R	0h	RAM Initialization status for GS5 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
4	INITDONE_GS4	R	0h	RAM Initialization status for GS4 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
3	INITDONE_GS3	R	0h	RAM Initialization status for GS3 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
2	INITDONE_GS2	R	0h	RAM Initialization status for GS2 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
1	INITDONE_GS1	R	0h	RAM Initialization status for GS1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
0	INITDONE_GS0	R	0h	RAM Initialization status for GS0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.

2.14.17.26 MSGxTEST Register (Offset = 70h) [reset = 0h]

MSGxTEST is shown in [Figure 2-245](#) and described in [Table 2-255](#).

Message RAM TEST Register

Figure 2-245. 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						R-0h	
7	6	5	4	3	2	1	0
RESERVED		TEST_CLA1TOCPU		TEST_CPUTOCLA1		TEST_CPUTOCPU	
R-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-255. 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	0h	Reserved
7-6	RESERVED	R	0h	Reserved
5-4	TEST_CLA1TOCPU	R/W	0h	Selects the different 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.
3-2	TEST_CPUTOCLA1	R/W	0h	Selects the different 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.
1-0	TEST_CPUTOCPU	R/W	0h	Selects the different modes for CPUTOCPU 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.

2.14.17.27 MSGxINIT Register (Offset = 72h) [reset = 0h]

MSGxINIT is shown in [Figure 2-246](#) and described in [Table 2-256](#).

Message RAM Init Register

Figure 2-246. 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	INIT_CPUTOC PU
R-0h			R-0h	R-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-256. MSGxINIT 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	INIT_CLA1TOCPU	R=0/W=1	0h	RAM Initialization control for CLA1TOCPU MSG RAM: 0: None. 1: Start RAM Initialization.
1	INIT_CPUTOCLA1	R=0/W=1	0h	RAM Initialization control for CPUTOCLA1 MSG RAM: 0: None. 1: Start RAM Initialization.
0	INIT_CPUTOCPU	R=0/W=1	0h	RAM Initialization control for CPUTOCPU MSG RAM: 0: None. 1: Start RAM Initialization.

2.14.17.28 MSGxINITDONE Register (Offset = 74h) [reset = 0h]

MSGxINITDONE is shown in [Figure 2-247](#) and described in [Table 2-257](#).

Message RAM InitDone Status Register

Figure 2-247. 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	INITDONE_CP UTCPU
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-257. 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.
1	INITDONE_CPUTOCLA1	R	0h	RAM Initialization status for CPUTOCLA1 MSG RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.
0	INITDONE_CPUTOCPU	R	0h	RAM Initialization status for CPUTOCPU MSG RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done.

2.14.18 ACCESS_PROTECTION_REGS Registers

Table 2-258 lists the memory-mapped registers for the ACCESS_PROTECTION_REGS. All register offset addresses not listed in Table 2-258 should be considered as reserved locations and the register contents should not be modified.

Table 2-258. ACCESS_PROTECTION_REGS Registers

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

2.14.18.1 NMAVFLG Register (Offset = 0h) [reset = 0h]

NMAVFLG is shown in [Figure 2-248](#) and described in [Table 2-259](#).

Non-Master Access Violation Flag Register

Figure 2-248. 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						R-0h	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	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-259. 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.
5	CLA1WRITE	R	0h	Non Master CLA1 Write Access Violation Flag: 0: No violation. 1: Access violation occurred.
4	CLA1READ	R	0h	Non Master CLA1 Read Access Violation Flag: 0: No violation. 1: Access violation occurred.
3	DMAWRITE	R	0h	Non Master DMA Write Access Violation Flag: 0: No violation. 1: Access violation occurred.
2	CPUFETCH	R	0h	Non Master CPU Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred.
1	CPUWRITE	R	0h	Non Master CPU Write Access Violation Flag: 0: No violation. 1: Access violation occurred.
0	CPUREAD	R	0h	Non Master CPU Read Access Violation Flag: 0: No violation. 1: Access violation occurred.

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

NMAVSET is shown in [Figure 2-249](#) and described in [Table 2-260](#).

Non-Master Access Violation Flag Set Register

Figure 2-249. 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						R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-260. NMAVSET 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=0/W=1	0h	0: No action. 1: CLA1 Fetch Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
5	CLA1WRITE	R=0/W=1	0h	0: No action. 1: CLA1 Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
4	CLA1READ	R=0/W=1	0h	0: No action. 1: CLA1 Read Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
3	DMAWRITE	R=0/W=1	0h	0: No action. 1: DMA Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
2	CPUFETCH	R=0/W=1	0h	0: No action. 1: CPU Fetch Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
1	CPUWRITE	R=0/W=1	0h	0: No action. 1: CPU Write Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..
0	CPUREAD	R=0/W=1	0h	0: No action. 1: CPU Read Access Violation Flag in NMAVFLG register will be set and interrupt will be generated if enabled..

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

NMAVCLR is shown in [Figure 2-250](#) and described in [Table 2-261](#).

Non-Master Access Violation Flag Clear Register

Figure 2-250. 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						R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-261. NMAVCLR 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=0/W=1	0h	0: No action. 1: CLA1 Fetch Access Violation Flag in NMAVFLG register will be cleared.
5	CLA1WRITE	R=0/W=1	0h	0: No action. 1: CLA1 Write Access Violation Flag in NMAVFLG register will be cleared.
4	CLA1READ	R=0/W=1	0h	0: No action. 1: CLA1 Read Access Violation Flag in NMAVFLG register will be cleared.
3	DMAWRITE	R=0/W=1	0h	0: No action. 1: DMA Write Access Violation Flag in NMAVFLG register will be cleared.
2	CPUFETCH	R=0/W=1	0h	0: No action. 1: CPU Fetch Access Violation Flag in NMAVFLG register will be cleared.
1	CPUWRITE	R=0/W=1	0h	0: No action. 1: CPU Write Access Violation Flag in NMAVFLG register will be cleared.
0	CPUREAD	R=0/W=1	0h	0: No action. 1: CPU Read Access Violation Flag in NMAVFLG register will be cleared.

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

NMAVINTEN is shown in [Figure 2-251](#) and described in [Table 2-262](#).

Non-Master Access Violation Interrupt Enable Register

Figure 2-251. 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						R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	CLA1FETCH	CLA1WRITE	CLA1READ	DMAWRITE	CPUFETCH	CPUWRITE	CPUREAD
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-262. NMAVINTEN 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/W	0h	0: CLA1 Non Master Fetch Access Violation Interrupt is disabled. 1: CLA1 Non Master Fetch Access Violation Interrupt is enabled.
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.
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.
3	DMAWRITE	R/W	0h	0: DMA Non Master Write Access Violation Interrupt is disabled. 1: DMA Non Master Write Access Violation Interrupt is enabled.
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.
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.
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.

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

NMCPURDAVADDR is shown in [Figure 2-252](#) and described in [Table 2-263](#).

Non-Master CPU Read Access Violation Address

Figure 2-252. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-263. 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.

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

NMCPUWRAVADDR is shown in [Figure 2-253](#) and described in [Table 2-264](#).

Non-Master CPU Write Access Violation Address

Figure 2-253. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-264. 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.

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

NMCPUFAVADDR is shown in [Figure 2-254](#) and described in [Table 2-265](#).

Non-Master CPU Fetch Access Violation Address

Figure 2-254. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-265. 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.

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

NMDMAWRAVADDR is shown in [Figure 2-255](#) and described in [Table 2-266](#).

Non-Master DMA Write Access Violation Address

Figure 2-255. 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
NMDMAWRAVADDR																															
R-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-266. 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.

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

NMCLA1RDAVADDR is shown in [Figure 2-256](#) and described in [Table 2-267](#).

Non-Master CLA1 Read Access Violation Address

Figure 2-256. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-267. 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.

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

NMCLA1WRAVADDR is shown in [Figure 2-257](#) and described in [Table 2-268](#).

Non-Master CLA1 Write Access Violation Address

Figure 2-257. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-268. 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.

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

NMCLA1FAVADDR is shown in [Figure 2-258](#) and described in [Table 2-269](#).

Non-Master CLA1 Fetch Access Violation Address

Figure 2-258. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-269. 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.

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

MAVFLG is shown in [Figure 2-259](#) and described in [Table 2-270](#).

Master Access Violation Flag Register

Figure 2-259. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-270. 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.
1	CPUWRITE	R	0h	Master CPU Write Access Violation Flag: 0: No violation. 1: Access violation occurred.
0	CPUFETCH	R	0h	Master CPU Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred.

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

MAVSET is shown in [Figure 2-260](#) and described in [Table 2-271](#).

Master Access Violation Flag Set Register

Figure 2-260. 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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-271. 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/W=1	0h	0: No action. 1: DMA Write Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled..
1	CPUWRITE	R=0/W=1	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled..
0	CPUFETCH	R=0/W=1	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled..

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

MAVCLR is shown in [Figure 2-261](#) and described in [Table 2-272](#).

Master Access Violation Flag Clear Register

Figure 2-261. 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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-272. 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/W=1	0h	0: No action. 1: DMA Write Access Violation Flag in MAVFLG register will be cleared.
1	CPUWRITE	R=0/W=1	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be cleared .
0	CPUFETCH	R=0/W=1	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be cleared.

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

MAVINTEN is shown in [Figure 2-262](#) and described in [Table 2-273](#).

Master Access Violation Interrupt Enable Register

Figure 2-262. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-273. 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.
1	CPUWRITE	R/W	0h	0: CPU Write Access Violation Interrupt is disabled. 1: CPU Write Access Violation Interrupt is enabled.
0	CPUFETCH	R/W	0h	0: CPU Fetch Access Violation Interrupt is disabled. 1: CPU Fetch Access Violation Interrupt is enabled.

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

MCPUFAVADDR is shown in [Figure 2-263](#) and described in [Table 2-274](#).

Master CPU Fetch Access Violation Address

Figure 2-263. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-274. 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.

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

MCPUWRAVADDR is shown in [Figure 2-264](#) and described in [Table 2-275](#).

Master CPU Write Access Violation Address

Figure 2-264. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-275. 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.

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

MDMAWRVADDR is shown in [Figure 2-265](#) and described in [Table 2-276](#).

Master DMA Write Access Violation Address

Figure 2-265. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-276. 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.

2.14.19 MEMORY_ERROR_REGS Registers

Table 2-277 lists the memory-mapped registers for the MEMORY_ERROR_REGS. All register offset addresses not listed in Table 2-277 should be considered as reserved locations and the register contents should not be modified.

Table 2-277. MEMORY_ERROR_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	UCERRFLG	Uncorrectable Error Flag Register		Go
2h	UCERRSET	Uncorrectable Error Flag Set Register	EALLOW	Go
4h	UCERRCLR	Uncorrectable Error Flag Clear Register	EALLOW	Go
6h	UCCPUREADDR	Uncorrectable CPU Read Error Address		Go
8h	UCDMAREADDR	Uncorrectable DMA Read Error Address		Go
Ah	UCCLA1READDR	Uncorrectable CLA1 Read Error Address		Go
20h	CERRFLG	Correctable Error Flag Register		Go
22h	CERRSET	Correctable Error Flag Set Register	EALLOW	Go
24h	CERRCLR	Correctable Error Flag Clear Register	EALLOW	Go
26h	CCPUREADDR	Correctable CPU Read Error Address		Go
2Eh	CERRCNT	Correctable Error Count Register		Go
30h	CERRTHRES	Correctable Error Threshold Value Register	EALLOW	Go
32h	CEINTFLG	Correctable Error Interrupt Flag Status Register		Go
34h	CEINTCLR	Correctable Error Interrupt Flag Clear Register	EALLOW	Go
36h	CEINTSET	Correctable Error Interrupt Flag Set Register	EALLOW	Go
38h	CEINTEN	Correctable Error Interrupt Enable Register	EALLOW	Go

2.14.19.1 UCERRFLG Register (Offset = 0h) [reset = 0h]

UCERRFLG is shown in [Figure 2-266](#) and described in [Table 2-278](#).

Uncorrectable Error Flag Register

Figure 2-266. 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	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-278. 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.
1	DMARDERR	R	0h	DMA Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during DMA read.
0	CPURDERR	R	0h	CPU Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during CPU read.

2.14.19.2 UCERRSET Register (Offset = 2h) [reset = 0h]

UCERRSET is shown in [Figure 2-267](#) and described in [Table 2-279](#).

Uncorrectable Error Flag Set Register

Figure 2-267. 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-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-279. UCERRSET 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=0/W=1	0h	0: No action. 1: CLA1 Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..
1	DMARDERR	R=0/W=1	0h	0: No action. 1: DMA Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..
0	CPURDERR	R=0/W=1	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled..

2.14.19.3 UCERRCLR Register (Offset = 4h) [reset = 0h]

UCERRCLR is shown in [Figure 2-268](#) and described in [Table 2-280](#).

Uncorrectable Error Flag Clear Register

Figure 2-268. 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-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-280. UCERRCLR 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=0/W=1	0h	0: No action. 1: CLA1 Read Error Flag in UCERRFLG register will be cleared.
1	DMARDERR	R=0/W=1	0h	0: No action. 1: DMA Read Error Flag in UCERRFLG register will be cleared .
0	CPURDERR	R=0/W=1	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be cleared.

2.14.19.4 UCCPUREADDR Register (Offset = 6h) [reset = 0h]

UCCPUREADDR is shown in [Figure 2-269](#) and described in [Table 2-281](#).

Uncorrectable CPU Read Error Address

Figure 2-269. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-281. 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.

2.14.19.5 UCDMAREADDR Register (Offset = 8h) [reset = 0h]

UCDMAREADDR is shown in [Figure 2-270](#) and described in [Table 2-282](#).

Uncorrectable DMA Read Error Address

Figure 2-270. UCDMAREADDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-282. 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 ECC/Parity error.

2.14.19.6 UCCLA1READDR Register (Offset = Ah) [reset = 0h]

UCCLA1READDR is shown in [Figure 2-271](#) and described in [Table 2-283](#).

Uncorrectable CLA1 Read Error Address

Figure 2-271. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-283. 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 ECC/Parity error.

2.14.19.7 CERRFLG Register (Offset = 20h) [reset = 0h]

CERRFLG is shown in [Figure 2-272](#) and described in [Table 2-284](#).

Correctable Error Flag Register

Figure 2-272. 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	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-284. 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.
1	DMARDERR	R	0h	DMA Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during DMA read.
0	CPURDERR	R	0h	CPU Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during CPU read.

2.14.19.8 CERRSET Register (Offset = 22h) [reset = 0h]

CERRSET is shown in [Figure 2-273](#) and described in [Table 2-285](#).

Correctable Error Flag Set Register

Figure 2-273. 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-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-285. CERRSET 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=0/W=1	0h	0: No action. 1: CLA1 Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..
1	DMARDERR	R=0/W=1	0h	0: No action. 1: DMA Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..
0	CPURDERR	R=0/W=1	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled..

2.14.19.9 CERRCLR Register (Offset = 24h) [reset = 0h]

CERRCLR is shown in [Figure 2-274](#) and described in [Table 2-286](#).

Correctable Error Flag Clear Register

Figure 2-274. 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-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-286. CERRCLR 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=0/W=1	0h	0: No action. 1: CLA1 Read Error Flag in CERRFLG register will be cleared.
1	DMARDERR	R=0/W=1	0h	0: No action. 1: DMA Read Error Flag in CERRFLG register will be cleared .
0	CPURDERR	R=0/W=1	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be cleared.

2.14.19.10 CCPUREADDR Register (Offset = 26h) [reset = 0h]

CCPUREADDR is shown in [Figure 2-275](#) and described in [Table 2-287](#).

Correctable CPU Read Error Address

Figure 2-275. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-287. 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.

2.14.19.11 CERRCNT Register (Offset = 2Eh) [reset = 0h]

CERRCNT is shown in [Figure 2-276](#) and described in [Table 2-288](#).

Correctable Error Count Register

Figure 2-276. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-288. 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.

2.14.19.12 CERRTHRES Register (Offset = 30h) [reset = 0h]

CERRTHRES is shown in [Figure 2-277](#) and described in [Table 2-289](#).

Correctable Error Threshold Value Register

Figure 2-277. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-289. 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.

2.14.19.13 CEINTFLG Register (Offset = 32h) [reset = 0h]

CEINTFLG is shown in [Figure 2-278](#) and described in [Table 2-290](#).

Correctable Error Interrupt Flag Status Register

Figure 2-278. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-290. 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.

2.14.19.14 CEINTCLR Register (Offset = 34h) [reset = 0h]

CEINTCLR is shown in [Figure 2-279](#) and described in [Table 2-291](#).

Correctable Error Interrupt Flag Clear Register

Figure 2-279. 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/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-291. 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/W=1	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be cleared.

2.14.19.15 CEINTSET Register (Offset = 36h) [reset = 0h]

CEINTSET is shown in [Figure 2-280](#) and described in [Table 2-292](#).

Correctable Error Interrupt Flag Set Register

Figure 2-280. 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/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-292. 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/W=1	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be set and interrupt will be generated if enabled.

2.14.19.16 CEINTEN Register (Offset = 38h) [reset = 0h]

CEINTEN is shown in [Figure 2-281](#) and described in [Table 2-293](#).

Correctable Error Interrupt Enable Register

Figure 2-281. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-293. 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.

2.14.20 ROM_WAIT_STATE_REGS Registers

Table 2-294 lists the memory-mapped registers for the ROM_WAIT_STATE_REGS. All register offset addresses not listed in Table 2-294 should be considered as reserved locations and the register contents should not be modified.

Table 2-294. ROM_WAIT_STATE_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	ROMWAITSTATE	ROM Wait State Configuration Register	EALLOW	Go

2.14.20.1 ROMWAITSTATE Register (Offset = 0h) [reset = 0h]

ROMWAITSTATE is shown in [Figure 2-282](#) and described in [Table 2-295](#).

ROM Wait State Configuration Register

Figure 2-282. ROMWAITSTATE 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							WSDISABLE
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-295. ROMWAITSTATE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	WSDISABLE	R/W	0h	0: ROM Wait State is enabled. CPU accesses to ROM are 1-wait. 1: ROM Wait State is disabled. CPU accesses to ROM are 0-wait.

2.14.21 FLASH_CTRL_REGS Registers

Table 2-296 lists the memory-mapped registers for the FLASH_CTRL_REGS. All register offset addresses not listed in Table 2-296 should be considered as reserved locations and the register contents should not be modified.

Table 2-296. FLASH_CTRL_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	FRDCNTL	Flash Read Control Register		Go
2h	FSPRD	Flash Read Margin Control Register		Go
1Eh	FBAC	Flash Bank Access Control Register		Go
20h	FBFALLBACK	Flash Bank Fallback Power Register		Go
22h	FBPRDY	Flash Bank Pump Ready Register		Go
24h	FPAC1	Flash Pump Access Control Register 1		Go
26h	FPAC2	Flash Pump Access Control Register 2		Go
28h	FMAC	Flash Module Access Control Register		Go
2Ah	FMSTAT	Flash Module Status Register		Go
180h	FRD_INTF_CTRL	Flash Read Interface Control Register		Go

2.14.21.1 FRDCNTL Register (Offset = 0h) [reset = F00h]

FRDCNTL is shown in [Figure 2-283](#) and described in [Table 2-297](#).

Flash Read Control Register

Figure 2-283. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-297. 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 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. An extra wait-state is automatically added when code is fetched (or data is read) from Bank1, even for prefetched data
7-0	RESERVED	R	0h	Reserved

2.14.21.2 FSPRD Register (Offset = 2h) [reset = 0h]

FSPRD is shown in [Figure 2-284](#) and described in [Table 2-298](#).

Flash Read Margin Control Register

Figure 2-284. FSPRD 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														RM1	RM0
R-0h														R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-298. FSPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	RM1	R/W	0h	0 Read Margin 1 mode is disabled 1 Read Margin 1 mode is enabled
0	RM0	R/W	0h	0 Read Margin 0 mode is disabled 1 Read Margin 0 mode is enabled

2.14.21.3 FBAC Register (Offset = 1Eh) [reset = Fh]

FBAC is shown in [Figure 2-285](#) and described in [Table 2-299](#).

Flash Bank Access Control Register

Figure 2-285. 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								VREADST							
R-0h																R/W-0h								R/W-Fh							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-299. 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.
7-0	VREADST	R/W	Fh	VREAD setup. VREAD is generated by the flash pump and used for flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable.

2.14.21.4 FBFALLBACK Register (Offset = 20h) [reset = 0h]

FBFALLBACK is shown in [Figure 2-286](#) and described in [Table 2-300](#).

Flash Bank Fallback Power Register

Figure 2-286. 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						BNKPWR0	
R-0h						R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-300. FBFALLBACK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1-0	BNKPWR0	R/W	0h	<p>Fall Back power mode</p> <p>00 Sleep (Sense amplifiers and sense reference disabled)</p> <p>01 Standby (Sense amplifiers disabled, but sense reference enabled)</p> <p>10 Reserved</p> <p>11 Active (Both sense amplifiers and sense reference enabled)</p> <p>Note: If the bank and pump are not in active mode and an access is made, the value of this register is automatically changed to active.</p>

2.14.21.5 FBPRDY Register (Offset = 22h) [reset = 0h]

FBPRDY is shown in [Figure 2-287](#) and described in [Table 2-301](#).

Flash Bank Pump Ready Register

Figure 2-287. 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							BANKRDY
R-0h							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-301. 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.
14-1	RESERVED	R	0h	Reserved
0	BANKRDY	R	0h	Bank Ready. This is a read-only register which allows software to determine if the bank 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 is not ready. 1 Bank is in active power mode and is ready for access.

2.14.21.6 FPAC1 Register (Offset = 24h) [reset = 640000h]

FPAC1 is shown in [Figure 2-288](#) and described in [Table 2-302](#).

Flash Pump Access Control Register 1

Figure 2-288. FPAC1 Register

31	30	29	28	27	26	25	24
RESERVED				PSLEEP			
R-0h				R/W-64h			
23	22	21	20	19	18	17	16
PSLEEP							
R/W-64h							
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-302. FPAC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-16	PSLEEP	R/W	64h	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.
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)

2.14.21.7 FPAC2 Register (Offset = 26h) [reset = 0h]

FPAC2 is shown in [Figure 2-289](#) and described in [Table 2-303](#).

Flash Pump Access Control Register 2

Figure 2-289. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-303. 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>

2.14.21.8 FMAC Register (Offset = 28h) [reset = 0h]

FMAC is shown in [Figure 2-290](#) and described in [Table 2-304](#).

Flash Module Access Control Register

Figure 2-290. FMAC 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												BANK			
R-0h												R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-304. FMAC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2-0	BANK	R	0h	0 BANKID. Controls the bank on which the Flash FSM operations will be performed. The value of this field will be 0 as there is only one bank per FMC.

2.14.21.9 FMSTAT Register (Offset = 2Ah) [reset = 0h]

FMSTAT is shown in [Figure 2-291](#) and described in [Table 2-305](#).

Flash Module Status Register

Figure 2-291. 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	ILA	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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-305. 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	ILA	R	0h	Illegal Address When set, indicates that an illegal address is detected. Three conditions can set illegal address flag. - Writing to a hole (un-implemented logical address space) within a flash bank. - Writing to an address location to an un-implemented flash space. - Input address for write is decoded to select a different bank from the bank ID register. - The address range does not match the type of FSM command.
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.
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.
9	RESERVED	R	0h	Reserved
8	BUSY	R	0h	When set, this bit indicates that a program, erase, or suspend operation is being processed.
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.

Table 2-305. FMSTAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 resumes.
5	INVDAT	R	0h	Invalid Data. When set, this bit indicates that the user attempted to program a "1" where a "0" was already present.
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.
3	VOLTSTAT	R	0h	Core Voltage Status. When set, this bit indicates that the core voltage generator of the pump power upply dipped below the lower limit allowable during a program or erase operation.
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

2.14.21.10 FRD_INTF_CTRL Register (Offset = 180h) [reset = 0h]

FRD_INTF_CTRL is shown in [Figure 2-292](#) and described in [Table 2-306](#).

Flash Read Interface Control Register

Figure 2-292. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-306. 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.
0	PREFETCH_EN	R/W	0h	Prefetch enable. 0 A value of 0 disables program cache and prefetch mechanism. 1 A value of 1 enables program cache and pre-fetch mechanism.

2.14.22 FLASH_ECC_REGS Registers

Table 2-307 lists the memory-mapped registers for the FLASH_ECC_REGS. All register offset addresses not listed in Table 2-307 should be considered as reserved locations and the register contents should not be modified.

Table 2-307. FLASH_ECC_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	ECC_ENABLE	ECC Enable		Go
2h	SINGLE_ERR_ADDR_LOW	Single Error Address Low		Go
4h	SINGLE_ERR_ADDR_HIGH	Single Error Address High		Go
6h	UNC_ERR_ADDR_LOW	Uncorrectable Error Address Low		Go
8h	UNC_ERR_ADDR_HIGH	Uncorrectable Error Address High		Go
Ah	ERR_STATUS	Error Status		Go
Ch	ERR_POS	Error Position		Go
Eh	ERR_STATUS_CLR	Error Status Clear		Go
10h	ERR_CNT	Error Control		Go
12h	ERR_THRESHOLD	Error Threshold		Go
14h	ERR_INTFLG	Error Interrupt Flag		Go
16h	ERR_INTCLR	Error Interrupt Flag Clear		Go
18h	FDATAH_TEST	Data High Test		Go
1Ah	FDATAL_TEST	Data Low Test		Go
1Ch	FADDR_TEST	ECC Test Address		Go
1Eh	FECC_TEST	ECC Test Address		Go
20h	FECC_CTRL	ECC Control		Go
22h	FOUTH_TEST	Test Data Out High		Go
24h	FOUTL_TEST	Test Data Out Low		Go
26h	FECC_STATUS	ECC Status		Go

2.14.22.1 ECC_ENABLE Register (Offset = 0h) [reset = Ah]

ECC_ENABLE is shown in [Figure 2-293](#) and described in [Table 2-308](#).

ECC Enable

Figure 2-293. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-308. 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.

2.14.22.2 SINGLE_ERR_ADDR_LOW Register (Offset = 2h) [reset = 0h]

SINGLE_ERR_ADDR_LOW is shown in [Figure 2-294](#) and described in [Table 2-309](#).

Single Error Address Low

Figure 2-294. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-309. 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.

2.14.22.3 SINGLE_ERR_ADDR_HIGH Register (Offset = 4h) [reset = 0h]

SINGLE_ERR_ADDR_HIGH is shown in [Figure 2-295](#) and described in [Table 2-310](#).

Single Error Address High

Figure 2-295. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-310. 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.

2.14.22.4 UNC_ERR_ADDR_LOW Register (Offset = 6h) [reset = 0h]

UNC_ERR_ADDR_LOW is shown in [Figure 2-296](#) and described in [Table 2-311](#).

Uncorrectable Error Address Low

Figure 2-296. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-311. 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.

2.14.22.5 UNC_ERR_ADDR_HIGH Register (Offset = 8h) [reset = 0h]

UNC_ERR_ADDR_HIGH is shown in [Figure 2-297](#) and described in [Table 2-312](#).

Uncorrectable Error Address High

Figure 2-297. UNC_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
UNC_ERR_ADDR_H																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-312. 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.

2.14.22.6 ERR_STATUS Register (Offset = Ah) [reset = 0h]

ERR_STATUS is shown in [Figure 2-298](#) and described in [Table 2-313](#).

Error Status

Figure 2-298. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-313. 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.
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.
16	FAIL_0_H	R	0h	Fail on 0. 0 Fail on 0 single bit error did not occur in lower 64bits of a 128-bit aligned address. 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 0. Cleared by writing a 1 to FAIL_0_H_CLR bit of ERR_STATUS_CLR register.
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.
1	FAIL_1_L	R	0h	Fail on 1. 0 Fail on 1 single bit error did not occur in lower 64bits of a 128-bit aligned address. 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.

Table 2-313. ERR_STATUS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 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 0. Cleared by writing a 1 to FAIL_0_L_CLR bit of ERR_STATUS_CLR register.</p>

2.14.22.7 ERR_POS Register (Offset = Ch) [reset = 0h]

ERR_POS is shown in [Figure 2-299](#) and described in [Table 2-314](#).

Error Position

Figure 2-299. 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					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-314. 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.
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.
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.
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.

2.14.22.8 ERR_STATUS_CLR Register (Offset = Eh) [reset = 0h]

ERR_STATUS_CLR is shown in [Figure 2-300](#) and described in [Table 2-315](#).

Error Status Clear

Figure 2-300. 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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h
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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-315. ERR_STATUS_CLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18	UNC_ERR_H_CLR	R=0/W=1	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.
17	FAIL_1_H_CLR	R=0/W=1	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.
16	FAIL_0_H_CLR	R=0/W=1	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.
15-3	RESERVED	R	0h	Reserved
2	UNC_ERR_L_CLR	R=0/W=1	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.
1	FAIL_1_L_CLR	R=0/W=1	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.
0	FAIL_0_L_CLR	R=0/W=1	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.

2.14.22.9 ERR_CNT Register (Offset = 10h) [reset = 0h]

ERR_CNT is shown in [Figure 2-301](#) and described in [Table 2-316](#).

Error Control

Figure 2-301. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-316. 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. In ECC logic test mode, ERR_CNT will keep incrementing for every cycle after a single bit error occurrence. So, in order to clear ERR_CNT in ECC logic test mode, ECC logic test mode has to be disabled prior to clearing the ERR_CNT using "Single Err Int Clear" bit.

2.14.22.10 ERR_THRESHOLD Register (Offset = 12h) [reset = 0h]

ERR_THRESHOLD is shown in [Figure 2-302](#) and described in [Table 2-317](#).

Error Threshold

Figure 2-302. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-317. 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.

2.14.22.11 ERR_INTFLG Register (Offset = 14h) [reset = 0h]

ERR_INTFLG is shown in [Figure 2-303](#) and described in [Table 2-318](#).

Error Interrupt Flag

Figure 2-303. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-318. 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.
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.

2.14.22.12 ERR_INTCLR Register (Offset = 16h) [reset = 0h]

ERR_INTCLR is shown in [Figure 2-304](#) and described in [Table 2-319](#).

Error Interrupt Flag Clear

Figure 2-304. 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_INTCLR	SINGLE_ERR_INTCLR
R-0h						R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-319. 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/W=1	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.
0	SINGLE_ERR_INTCLR	R=0/W=1	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.

2.14.22.13 FDATAH_TEST Register (Offset = 18h) [reset = 0h]

FDATAH_TEST is shown in [Figure 2-305](#) and described in [Table 2-320](#).

Data High Test

Figure 2-305. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-320. 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 blocks in ECC test mode.

2.14.22.14 FDATA_TEST Register (Offset = 1Ah) [reset = 0h]

FDATAL_TEST is shown in [Figure 2-306](#) and described in [Table 2-321](#).

Data Low Test

Figure 2-306. FDATA_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
FDATAL																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-321. FDATA_TEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	FDATAL	R/W	0h	Low double word of selected 64-bit data. User-configurable bits 31:0 of the selected data blocks in ECC test mode.

2.14.22.15 FADDR_TEST Register (Offset = 1Ch) [reset = 0h]

FADDR_TEST is shown in [Figure 2-307](#) and described in [Table 2-322](#).

ECC Test Address

Figure 2-307. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-322. 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.
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.
2-0	RESERVED	R	0h	Reserved

2.14.22.16 FECC_TEST Register (Offset = 1Eh) [reset = 0h]

FECC_TEST is shown in [Figure 2-308](#) and described in [Table 2-323](#).

ECC Test Address

Figure 2-308. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-323. 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.

2.14.22.17 FECC_CTRL Register (Offset = 20h) [reset = 0h]

FECC_CTRL is shown in [Figure 2-309](#) and described in [Table 2-324](#).

ECC Control

Figure 2-309. 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/W=1-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-324. 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/W=1	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.
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.
0	ECC_TEST_EN	R/W	0h	ECC test mode enable. 0 ECC test mode disabled 1 ECC test mode enabled

2.14.22.18 FOUTH_TEST Register (Offset = 22h) [reset = 0h]

FOUTH_TEST is shown in [Figure 2-310](#) and described in [Table 2-325](#).

Test Data Out High

Figure 2-310. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-325. 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.

2.14.22.19 FOUTL_TEST Register (Offset = 24h) [reset = 0h]

FOUTL_TEST is shown in [Figure 2-311](#) and described in [Table 2-326](#).

Test Data Out Low

Figure 2-311. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-326. 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.

2.14.22.20 FECC_STATUS Register (Offset = 26h) [reset = 0h]

FECC_STATUS is shown in [Figure 2-312](#) and described in [Table 2-327](#).

ECC Status

Figure 2-312. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 2-327. 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).
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.
1	UNC_ERR	R	0h	Test mode ECC double bit error. When 1 indicates that the ECC test resulted in an uncorrectable bit error.
0	SINGLE_ERR	R	0h	Test mode ECC single bit error. When 1 indicates that the ECC test resulted in a single bit error.

ROM Code and Peripheral Booting

This chapter describes the booting functionality of the 2837xD subsystems.

Topic	Page
3.1 Introduction	520
3.2 Device Boot Philosophy	520
3.3 Device Clocking on Power-up and Boot Time	520
3.4 Clock Configurations in Boot ROM	520
3.5 Faster Flash Power Up	521
3.6 Boot ROM DCSM init Sequence.....	521
3.7 Device Calibration on CPU1	522
3.8 CPU1 Boot ROM Procedure	522
3.9 CPU2 Boot ROM Procedure	523
3.10 Boot Modes Supported on CPU1	523
3.11 Boot Modes Supported on CPU2	528
3.12 CPU1 Boot ROM Flow Chart	530
3.13 CPU2 Boot ROM Flow Chart.....	533
3.14 Boot ROM Reset Causes and Handling	534
3.15 Exceptions and Interrupts handling	534
3.16 CPU1 OTP Boot Configure Word	535
3.17 Boot ROM Status information	535
3.18 Boot and IPC Commands	537
3.19 Device Boot Process Timeline Diagram.....	541
3.20 Boot ROM GPIO Configurations:	542
3.21 Boot ROM Memory Map	543
3.22 CPU1 and CPU2 ROM REVISION Information	544
3.23 RAM and Flash Usage and Application Entry Points	544
3.24 ROM Wait States	545
3.25 Device Boot Modes Description	546
3.26 CLA Data ROM	569
3.27 Secure ROM Contents	571

3.1 Introduction

The purpose of this document is to mainly state the functions and features of boot ROM code and also define the contents and allocated memory map of other ROM(s) in the device.

NOTE: This chapter is in the formative phase of development. Future revisions will be edited for clarity.

3.2 Device Boot Philosophy

The state of the system after an external reset, power up or watchdog reset is as follows:

- CPU1 starts execution from CPU1 Boot ROM
- CPU2 is held in reset

The CPU1 will be the master controller and will also control the boot process. The CPU2 will go through its own boot procedure, but under the control of the CPU1, except when CPU2 is set to boot-to-flash.

Below are the resources used during the boot process:

CPU1 Boot ROM	Code execution starts from this ROM after reset.
CPU1 OTP	This OTP contains the device configuration; security configuration and analog trim information that is used during boot process.
CPU2 OTP	This OTP contains device calibration stuff for the CPU2
CPU BOOT ROM	Code execution starts from this ROM for the CPU2 after being released from reset by the CPU1.
C1toC and C2toC1 IPC	Registers used by CPU1 to communicate to the CPU2 during boot process and vice-versa

The CPU1 has no direct access to registers on the CPU2 side. Therefore, IPC registers are used to allow the CPU1 to communicate to the CPU2 during boot process.

Each core polls for IPC commands that it supports and performs simple actions as indicated below. IPC commands supported by each core will be explained in detail later in this document.

- Perform 8/16/32-bit read, write, read-modify-write operations to specified address locations
- Branch to specified address and does not return (supported only on CPU2).
- Call a function and return (supported only on CPU2)

In the IPC, a command (xIPCCOM), address (xIPCADDR), data write (xIPCDATAW) and data read (xIPC DATAR) register is provided that is initialized by the CPUx before starting a communication.

In addition, the xBOOTSTS and xBOOTMODE registers are used by the CPU1 and CPU2 to determine the current state and also synchronize actions during the boot process.

3.3 Device Clocking on Power-up and Boot Time

Device (both CPU1 and CPU2) boots up with INTOSC2 as the default clock source unless there is a missing clock detected at power up or right after device reset. In this case, the device will boot up with INTOSC1 as the clock source.

3.4 Clock Configurations in Boot ROM

This section describes the clock configurations done in boot ROM. All the below configurations are done only for a POR or XRSn or HIBRESETn reset types. For all other resets boot ROM starts executing with the clocks that are already setup before reset.

3.4.1 CPU1 Boot ROM Clocking Configuration

PLL is bypassed by default on POR or XRSn or HIBRESETn and default SYSDIVSEL is set to /4 by design.

To assist with fast boot time response, Boot ROM will:

- Set the SYSDIVSEL divider to a /1
- Power up PLL and set an integer multiplier. Configure flash wrapper for faster flash power-up (see [Section 3.5](#))
- Check PLL Lock status and put PLL output in clock path

PLL multiplier and divider decisions are based on the below reasoning:

- Need to use PLL for better boot time response
- PLL Fout range is 150-550 MHz, the multiplier chosen should make sure of that by taking the max/min clock frequency that could be available during worst case and best case.

Note that CPU1 Boot ROM writes to the PLL multiplier and enables PLL before accessing Flash, the PLL lock time of 125uS is more than the Flash wake up time, so by the time Flash is powered up and ready PLL would have locked. After Flash is powered up CPU1 Boot ROM looks at PLL LOCK Status and puts the PLLOUTPUT in clock path, else PLLOUTPUT is not used in clock path.

NOTE: PLL if used during boot process, will be bypassed by Boot ROM before branching to the application.

3.4.2 CPU2 Boot ROM Clocking Configuration

There is no clock configuration done in CPU2 Boot ROM.

3.5 Faster Flash Power Up

This section describes the optimizations done in boot ROM for RWAIT and PSLEEP values in Flash wrapper for faster Flash power up during boot. The modifications/optimizations are done only for POR, XRSn or HIBRESETn reset types.

The first access to Flash after POR/XRSn/HIBRESETn will wake up the Flash and until Flash wakes up or is ready, CPU will stall.

CPU1 Boot ROM modifies PSLEEP to 0x120 and RWAIT to 0x3 before accessing Flash. After Flash is powered up the RWAIT is set back to 0xF and PSLEEP to 0x860.

CPU2 Boot ROM – the RWAIT is modified to 0x3 before accessing Flash and set back to 0xF after the bank powers up. PSLEEP is not modified because by the time CPU2 Boot ROM accesses its own Flash bank the pump is already powered up by CPU1 Boot ROM. So there is no need of modifying PSLEEP.

3.6 Boot ROM DCSM init Sequence

3.6.1 CPU1 DCSM init Sequence

- Read triple link pointers
- Decode link pointer
- Read Zx BOOT MODE
- Read SECDC (0x703F0, TI-reserved register) in TI OTP
- Read the Zx DCSM block

3.6.2 CPU2 DCSM init Sequence

- Read triple link pointers
- Decode link pointer
- Read Zx BOOT MODE
- Read SECDC (0x703F0, TI-reserved register) in TI OTP
- Read the Zx DCSM block

3.7 Device Calibration on CPU1

This applies to CPU1 only.

The following modules require TI factory supplied trims to be loaded into the module trim registers in order to meet datasheet specifications:.

- Buffered DAC
- 0-pin oscillators
- ADC

These trims are cleared by a reset pulse on XRSn and when entering the hibernate low power mode. When the reset pulse is complete or the device exits hibernate, the boot ROM executes and loads the following trims directly:

- 0-pin oscillator trim

The boot ROM also calls the device calibration function in TI OTP. This function populates the following trims:

- Buffered DAC offset trim
- ADC bandgap trim
- ADC offset trim (see note below)
- ADC linearity trim (see note below)

NOTE: The ADC module requires a different offset trim for different signal mode and resolution settings. The ADC module requires a different linearity trim for different resolution settings. Calling the `AdcSetMode` function to set the desired resolution and signal mode will also ensure the correct trims are populated. The device calibration function will call the `AdcSetMode` function with 12-bit, single-ended arguments. In order to set the ADC in a different signal mode or resolution, use the `AdcSetMode` function to write those settings to ensure that the correct trim is loaded.

3.8 CPU1 Boot ROM Procedure

CPU1 Boot ROM is executed each time CPU1 core is reset. CPU1 acts as master system in the device controlling the overall boot process on CPU2 system. If booting from one of the bootable peripherals then CPU1 Boot ROM gets the application code to be run on CPU1 and loads into device RAMs (M0, M1,LS0-LS5, D0, D1, and GS0-GS15 shared RAM) and starts the CPU1 application. It is the responsibility of CPU1 user application to either move the code from CPU1 RAMs to CPU2 RAM(s) or kick start its own peripheral loaders using CPU2 Boot ROM IPC.

CPU1 Boot ROM will follow this procedure:

- Program device configuration registers on the device from OTP.
- CPU1 Boot ROM will execute DCSM and OTPJTAGLOCK sequences
- Check for FUSEERR register to check for any errors, and take appropriate action.
- Do a RAM-INIT of all CPU1 RAM(s) to initialize ECC/PARITY on the POR or after the HIBERNATE Reset.
- Handle the exceptions/NMI which are enabled by default on reset. How CPU1 Boot ROM handles these is explained later in the document.
- Bring CPU2 out of RESET as part of device initialization process and irrespective of whether it is booting from Flash or not.
- Poll BOOTMODE GPIO to boot from Flash/SRAM/peripherals after system initialization
- Update the boot status of core to application a specific RAM location. CPU1 applications can read this RAM location and find out more about device boot status or health status. A description of boot status to app is explained later in this document.
- If device is set to boot from Flash, then CPU1 Boot ROM just starts the application in CPU1 Flash. It is the responsibility of application in Flash to program/start the CPU1 core. CPU1 Boot ROM just does the needful to make sure both CPU2 Boot ROM is ready to accept BOOTMODE commands and IPC

commands from Master system.

- CPU1 Boot ROM will not try to start any application on CPU2 system. Only CPU1 application is allowed to start the application on CPU2 system.
- CPU1 Boot ROM when in WAIT BOOT MODE supports predefined IPC commands to allow for easier test application development on CPU2.

3.9 CPU2 Boot ROM Procedure

On power-up CPU2 system will be held in reset until CPU1 Boot ROM brings it out of reset, when CPU2 is out of reset it will start executing code from CPU2 Boot ROM memory. Each time CPU2 is reset CPU2 Boot ROM is executed and depending on the reset cause CPU2 Boot ROM will take actions as explained in further below in this document.

- CPU2 Boot ROM on start up initializes CPU2 sub system, performs RAM-INIT as per the RESC, installs PIE interrupt handlers to serve IPC commands from Master system and goes to IDLE mode.
- CPU2 Boot ROM is allowed to boot to Flash when boot mode is set to boot-to-flash, to allow Checker Micro kind of applications on CPU2.
- CPU1 application before starting CPU2 peripheral boot, is supposed to configure all the IOs and the peripheral ownership after checking if the peripheral is existing in the current device part number (DCx register check). CPU2 cannot read any of these status so it will simply try to boot from the peripheral or IO.

Please refer to the flow chart for the CPU2 Boot ROM procedure.

3.10 Boot Modes Supported on CPU1

Device allows users to download their application on to RAM using the following boot mode configurations. Boot loader uses boot mode GPIO to know from which peripheral to boot from.

CPU1 Boot ROM is responsible to decode the boot mode by reading the boot mode GPIO as defined in the below table. After CPU1 Boot ROM loads/starts CPU1-Application in CPU1-Flash/RAM it is the responsibility of the application to start the user application on CPU2. CPU2 Boot ROM supported both peripheral boot loading and boot from Flash options using IPC commands.

CPU1 Boot ROM boot loader supports booting from:

- CPU1 Flash
- SCI
- SPI
- I2C
- CPU1-RAM
- CAN
- USB

NOTE: All the peripheral boots supported on CPU1/CPU2 are on primary peripherals (SCIA, SPIA, I2CA, CANA, and so on). So whenever SCIBOOT is referred to in this chapter, it actually means SCIA Boot or Boot on SCIA port. The same is applied for other peripheral boots as well.

Peripheral boot loading will be supported on CPU2 Boot ROM but CPU1 application will have to configure the peripheral and IOs appropriately before sending the boot mode IPC command.

Table 3-1. Device Boot Mode – Decoded by CPU1

Mode No	CPU1 Boot Mode	CPU2 Boot Mode	TRSTn	GPIO72 (boot mode pin 1)	GPIO84 (boot mode pin 0)	Supported in Version 1.0
0	Parallel IO	Boot from Master	0	0	0	Yes
1	SCI Mode	Boot from Master	0	0	1	Yes
2	Wait Boot Mode	Boot from master	0	1	0	Yes
3	Get Mode	Boot from Master	0	1	1	Yes
4-7	EMU Boot Mode (Emulator Connected)	Boot from Master	1	X	X	Yes

Bootting from device COM peripherals will follow the same format and protocol as supported in Piccolo class devices.

- Mode EMU (modes 4 thru 7): When TRSTn is HIGH, debugger is connected and the boot ROM determines the boot mode from the first two locations (reset vector) in the PIE vector table.

The definition of EMU_BMODE and EMU_KEY is as below:- 32 bits from 0xD00. This definition is similar to the Zx-BOOTCTRL register definition in OTP and the reason for defining this similar to the Zx-BOOTCTRL is to allow users emulate the device boot with debugger connected before they go ahead and program the OTP with final values.

Table 3-2. EMU_BOOTCTRL Definition: - @ 0xD00 for CPU1 (32 bit word)

Bits	Name	Type	Reset	Description
7:0	EMU_KEY	R/W	X	Use 0x5A to indicate validity of this location values.
15:8	EMU_BMODE	R/W	X	Use this field to define up to 256 boot modes
23:1 6	EMU_BOOTPIN0	R/W	X	0 -> Pick the default boot pin-0
				1 -> Pick GPIO0 as boot pin-0
				2 -> Pick GPIO1 as boot pin-0
			
				255 -> Pick GPIO255 as boot pin-0
31:2 4	EMU_BOOTPIN1	R/W	X	0 -> Pick the default boot pin-1
				1 -> Pick GPIO0 as boot pin-1
				2 -> Pick GPIO1 as boot pin-1
			
				255 -> Pick GPIO255 as boot pin-1

EMU_BMODE/EMU_KEY will be populated with the boot mode by the boot ROM when powering up with TRSTn = 0.

Above locations can also be initialized manually by the user through the debugger. The value in EMU_KEY must also be initialized to indicate EMU_BMODE is valid. To do this, the boot ROM, or user, will write 0x5A to EMU_KEY. This is to help avoid treating random data in EMU_BMODE as a boot mode. An invalid key or mode will result in “wait” which is the safest mode.

The EMU_BOOTPIN0 and EMU_BOOTPIN1 are provided for users to emulate the device behavior with debugger connected before programming the OTP with final values in case they are choosing a different boot mode pins other than the factory chosen ones. If a user is choosing the factory default ones then they can leave EMU_BOOTPIN0 and EMU_BOOTPIN1 un-programmed, the boot ROM decides whether to look at these values or not based on the EMU_BMODE value as per the table given below.

Table 3-3. Emulation Boot (TRSTn == 1)

EMU_KEY	EMU_BMODE	EMU_BOOTPIN1	EMU_BOOTPIN0	Realized Boot Mode	CPU1/CPU2
!= 0x5A	Don't Care	Don't Care	Don't Care	WAIT BOOT	Both
0x5A	0xFE	Valid (pin decode as per table 13-2)	Valid (pin decode as per table 13-2)	Boot as per EMU_BOOTPIN0 and EMU_BOOTPIN1	CPU1 only
		0	0	Parallel BOOT	
		0	1	SCI BOOT0	
		1	0	WAIT BOOT	
		1	1	GET MODE (read OTP Zx-BOOTCTRL)	
	0xFF	Don't Care	Don't Care	Emulate CPU1 Stand alone boot (TRSTn == 0)	Both
	0x00	Don't Care	Don't Care	Parallel BOOT	Both
	0x01	Don't Care	Don't Care	SCI BOOT0	Both
	0x02	Don't Care	Don't Care	WAIT BOOT	Both
	0x03	Don't Care	Don't Care	GET MODE (read OTP Zx-BOOTCTRL)	Both
	0x04	Don't Care	Don't Care	SPI BOOT0	Both
	0x05	Don't Care	Don't Care	I2C BOOT0	Both
	0x07	Don't Care	Don't Care	CAN BOOT0	Both
	0x0A	Don't Care	Don't Care	RAM BOOT	Both
	0x0B	Don't Care	Don't Care	FLASH BOOT	Both
	0x0C	Don't Care	Don't Care	USB BOOT	CPU1 ONLY
	0x81	Don't Care	Don't Care	SCI BOOT1	CPU1 ONLY
	0x84	Don't Care	Don't Care	SPI BOOT1	CPU1 ONLY
	0x85	Don't Care	Don't Care	I2C BOOT1	CPU1 ONLY
	0x87	Don't Care	Don't Care	CAN BOOT1	CPU1 ONLY
	0x47	Don't Care	Don't Care	CAN BOOT TEST MODE0	Both
	0xC7	Don't Care	Don't Care	CAN BOOT TEST MODE1	CPU1 ONLY
	Other	Don't Care	Don't Care	WAIT BOOT	Both

The user can modify EMU_KEY and EMU_BMODE to change the boot mode via the debugger. A typical debug sequence would be:

1. Power the device, TRSTn = 0.

Configure the chosen boot mode pins for mode 0 – “wait”. This mode emulates a wait-in-reset mode. The boot ROM will loop, allowing the debugger a chance to connect without the eCSL tripping. This will also keep any application code from starting. During this time, the boot ROM will also populate EMU_KEY with 0x5A and EMU_BMODE with WAIT_BOOT.

2. Connect the debugger, TRSTn goes high.

Via the debugger change EMU_BMODE to the desired boot mode. For example, during debug you may want to use “boot to RAM”. To achieve this, load EMU_BMODE with 0x000A.

The EMU_KEY should already be initialized by the boot ROM to 0x5A.

NOTE: TI will be updating the header file peripheral examples that run from SARAM to load EMU_BMODE with “RAM_BOOT” when the code is loaded. In this case, step 2 will simply be to load the code.

3. Perform a debugger reset and run.

4. The boot ROM will detect TRSTn high and will use EMU_KEY and EMU_BMODE to determine the boot mode.

If either EMU_KEY or EMU_BMODE is invalid, then the mode will be 'wait'.

NOTE: The behavior of emulators with regards to TRSTn differs. This must be documented clearly.

Some emulators will pull TRSTn high only when Code Composer Studio™ is in a connected state. For these emulators, if Code Composer Studio is disconnected, TRSTn will return to a low state. With Code Composer Studio disconnected, boot mode 0 and boot mode 1 pins (ref. to above boot mode table) will be used to determine the boot mode. On these emulators, this is true even if the emulator pod is physically connected.

Some emulators will pull TRSTn high when Code Composer Studio connects and leave it high as long as the power sense pin is active. TRSTn will remain high even after CCS disconnects. For these emulators, the EMU mode stored in RAM will be used unless the target is power cycled, causing the state of TRSTn to reset back to a low state.

The following modes are available when TRSTn = 0.

In this case the boot mode pins are used to determine the boot mode of the device. The factory default boot mode pins are GPIO84 (boot mode pin 0) and GPIO 72 (boot mode pin 1). You are allowed to choose a different boot mode pin combination by programming the Zx-BOOTCTRL register as shown in [Table 3-4](#).

Table 3-4. BOOTCTRL Register Bit Definition for CPU1⁽¹⁾

Bits	Name	Type	Reset	Description
7:0	OTP_KEY	R	0	Use 0x5A to indicate validity of this register.
15:8	OTP_BMODE	R	0	Use this field to define up to 256 boot modes
23:1 6	OTP_BOOTPIN0	R	0	0 -> Pick the default boot pin-0 1 -> Pick GPIO0 as boot pin-0 2 -> Pick GPIO1 as boot pin-0 255 -> Pick GPIO245 as boot pin-0
31:2 4	OTP_BOOTPIN1	R	0	0 -> Pick the default boot pin-1 1 -> Pick GPIO0 as boot pin-1 2 -> Pick GPIO1 as boot pin-1 255 -> Pick GPIO254 as boot pin-1

⁽¹⁾ Refer to the DCSM chapter for the address of this location in the user OTP.

Table 3-5. BOOTCTRL Register Bit Definition for CPU2⁽¹⁾

Bits	Name	Type	Reset	Description
7:0	OTP_KEY	R	0	Use 0x5A to indicate validity of this register.
15:8	OTP_BMODE	R	0	Use this field to define up to 256 boot modes
23:1 6	Reserved	R	0	Reserved for future use
31:2 4	Reserved	R	0	Reserved for future use

⁽¹⁾ Refer to the DCSM chapter for the address of this location in the user OTP.

- Mode 2: "wait" This is to emulate a wait-in-reset mode. This is used when bringing up a debugger on a device with the CSM passwords programmed. This mode writes WAIT_BOOT to EMU_BMODE. This mode enables the PIE on CPU1 and installs C2toC1IPC interrupt handler and loops infinitely.
- Mode 1: SCI – a common method for programming devices. This mode writes SCI_BOOT to the EMU_BMODE.

- Mode 0: Parallel I/O 8-bit. This will be used by production programmers. This mode writes PARALLEL_IO_BOOT to EMU_BMODE.
- Mode 3: Get Mode. On a fresh, un-programmed, device this mode will always boot to Flash. On a programmed device, the user can program the values into Zx-BOOTCTRL location in the user OTP to change the behavior. This mode writes GET_BOOT to EMU_BMODE.

The values used by the Get_mode() function are as follows:

Note that GETMODE can be entered when emulator is connected as well as will be shown further below in the document. The boot mode decoding for Get Mode when emulator is connected is as shown below.

Table 3-6. Get Mode Decoding on CPU1

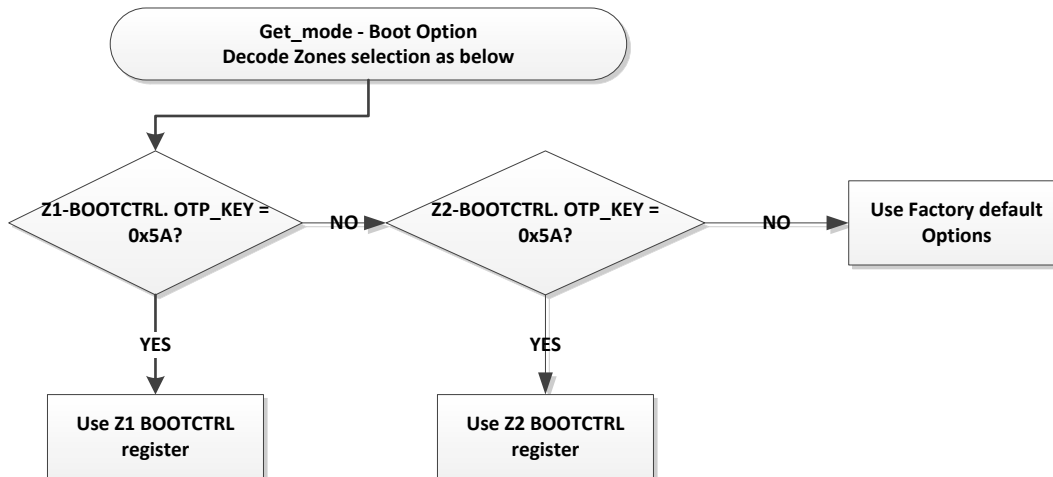
OTP_KEY	OTP_BMODE	Realized Boot Mode When TRSTn == 0	Realized Boot Mode When TRSTn == 1
!= 0X5A	Don't Care	FLASH BOOT	FLASH BOOT
0x5A	0x00	Parallel BOOT	Parallel BOOT
	0x01	SCI BOOT0	SCI BOOT0
	0x02	WAIT BOOT	WAIT BOOT
	0x04	SPI BOOT	SPI BOOT0
	0x05	I2C BOOT	I2C BOOT0
	0x07	CAN BOOT	CAN BOOT0
	0x0A	RAM BOOT	RAM BOOT
	0x0B	FLASH BOOT	FLASH BOOT
	0x0C	USB BOOT	USB BOOT
	0x81	SCI BOOT1	SCI BOOT1
	0x84	SPI BOOT1	SPI BOOT1
	0x85	I2C BOOT1	I2C BOOT1
	0x87	CAN BOOT1	CAN BOOT1
	0x47	CAN BOOT TEST0 IOs (special mode which tx frames for bit rate sync up)	CAN BOOT TEST1 (special mode which tx frames for bit rate sync up)
	0xC7	CAN BOOT TEST1 (special mode which tx frames for bit rate sync up)	CAN BOOT TEST1 (special mode which tx frames for bit rate sync up)
	Other	FLASH BOOT	WAIT BOOT

Table 3-7. Get Mode Decoding on CPU2

OTP_KEY	OTP_BMODE	Realized Boot Mode When TRSTn == 0	Realized Boot Mode When TRSTn == 1	Special Cases?
!= 0X5A	Don't Care	WAIT BOOT	WAIT BOOT	N/A
0x5A	0x0A	RAM BOOT	RAM BOOT	Only in HIBRESC case, all other reset causes this will default to WAIT BOOT
	0x0B	FLASH BOOT	FLASH BOOT	N/A
	Other	WAIT BOOT	WAIT BOOT	N/A

On this device there are two CSM Zones, Z1 and Z2 each with their own copy of Zx-BOOTCTRL register, boot ROM uses below procedure to identify which one to choose. By default if Z1 is programmed then Z1 is given preference, else Z2. If neither are programmed then factory default options are chosen.

Figure 3-1. BOOTCTRL Selection (applies to both the CPUs)



Mode 0, 1, 2 and 3 result in the following boot modes when TRSTn = 0:

Table 3-8. Stand Alone Boot Modes With TRSTn = 0 (on CPU1)

TRSTn	EMU_BOOTCTRL	OTP_KEY	OTP_BOOTPIN1	OTP_BOOTPIN0	OTP_BMODE	Boot Mode
0	Don't care	!=0x5A	default GPIO72	default GPIO84		
			0	0	Don't care	PARALLEL BOOT
			0	1	Don't care	SCI BOOT
			1	0	Don't care	WAIT BOOT
			1	1	Don't care (no key match)	GET BOOT → FLASH BOOT
		0x5A	User Chosen Boot Pin 1	User Chosen Boot pin 0		
			0	0	Don't care	PARALLEL BOOT
			0	1	Don't care	SCI BOOT
			1	0	Don't care	WAIT BOOT
			1	1	GET MODE (as per Table 3-6) ⁽¹⁾	

⁽¹⁾ Get Mode indicates the boot mode was derived from the values programmed in the OTP_KEY and OTP_BMODE locations, read by boot ROM init process and loaded into BOOTMODE register.

3.11 Boot Modes Supported on CPU2

On CPU2, both EMU-BOOT mode (TRSTn ==1) and stand-alone boot modes (TRSTn ==0) are supported similar to CPU1 except that CPU2 doesn't support decoding of any boot mode GPIO pins and instead reads the OTP Zx-BOOTCTRL register to see if boot-to-flash is selected and if not selected then IPC interrupt is enabled and CPU is put to IDLE mode.

When CPU2 Boot ROM is run with emulator connected, similar to CPU1, the EMUBOOTCTRL location (0xD00 – 32 bits) is used to by boot ROM to know how to boot.

Table 3-9. EMU_BOOTCTRL Definition: - @ 0xD00 for CPU2

Bits	Name	Type	Reset	Description
7:0	EMU_KEY	R	0	Use 0x5A to indicate validity of this register.
15:8	EMU_BMODE	R	0	Use this field to define up to 256 boot modes

Table 3-9. EMU_BOOTCTRL Definition: - @ 0xD00 for CPU2 (continued)

Bits	Name	Type	Reset	Description
23:16	Reserved	R	0	0 -> Pick the default boot pin-0
31:24	Reserved	R	0	0 -> Pick the default boot pin-1

Below are the EMU Boot modes supported by CPU2, note that there is no USB boot support on CPU2.

Table 3-10. Emulation Boot (TRSTn == 1) – CPU2

EMU_KEY	EMU_BMODE	EMU_BOOTPIN1	EMU_BOOTPIN0	Realized Boot Mode
!= 0X5A	Don't Care	Don't Care	Don't Care	WAIT BOOT
0x5A	0xFF	Don't Care	Don't Care	Emulate CPU2 Stand alone boot (TRSTn == 0)
	0x00	Don't Care	Don't Care	Parallel BOOT
	0x01	Don't Care	Don't Care	SCI BOOT
	0x02	Don't Care	Don't Care	WAIT BOOT
	0x03	Don't Care	Don't Care	GET MODE (as shown in Table 3-7)
	0x04	Don't Care	Don't Care	SPI BOOT
	0x05	Don't Care	Don't Care	I2C BOOT
	0x06	Don't Care	Don't Care	WAIT BOOT
	0x07	Don't Care	Don't Care	CAN BOOT
	0x0A	Don't Care	Don't Care	RAM BOOT
	0x0B	Don't Care	Don't Care	FLASH BOOT
	Other	Don't Care	Don't Care	WAIT BOOT

Stand alone boot on CPU2 support is shown in [Table 3-11](#).

Table 3-11. Stand Alone Boot Options on CPU2

TRSTn	EMU_BOOTCTRL	OTP_KEY	OTP_BMODE	Realized Boot Mode
0	Don't care	!=0x5A	Don't care	WAIT BOOT
		0x5A	XXXX	As shown in GET MODE Table 3-7

WAIT MODE on CPU2:

Enable C1 to C2 PIE interrupt handler and enter IDLE mode with WDOG disabled.

3.12 CPU1 Boot ROM Flow Chart

Figure 3-2. Flow Chart (part 1)

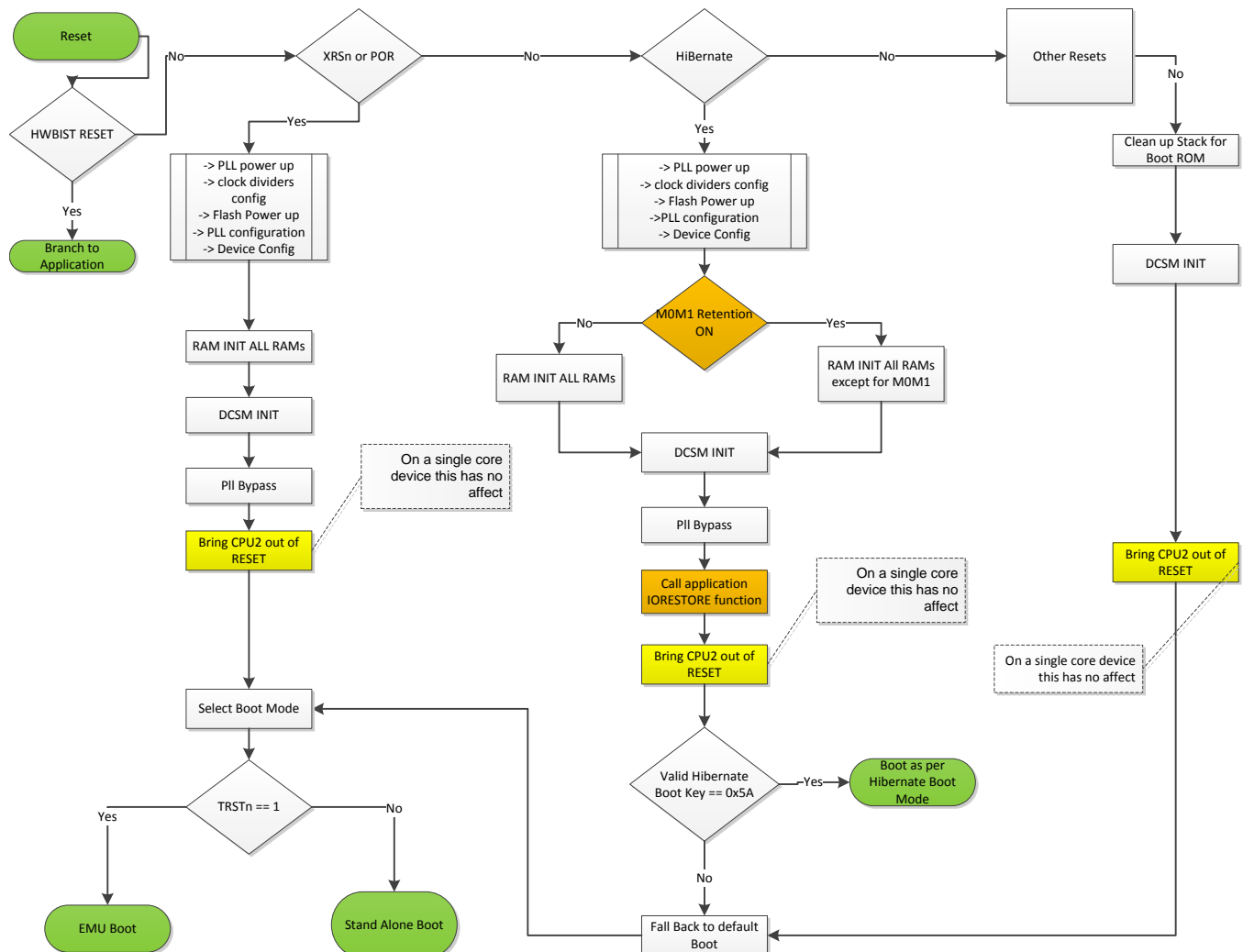


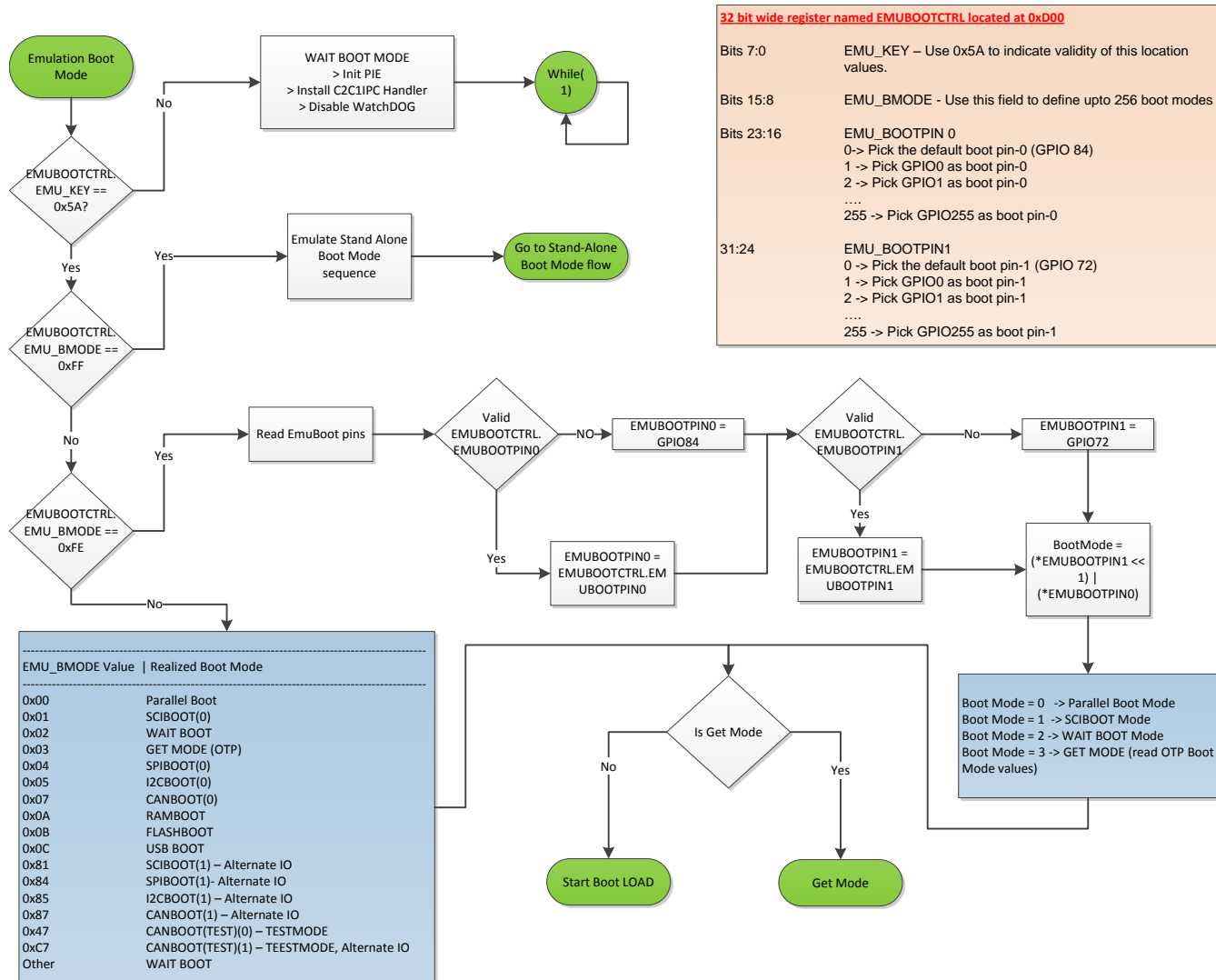
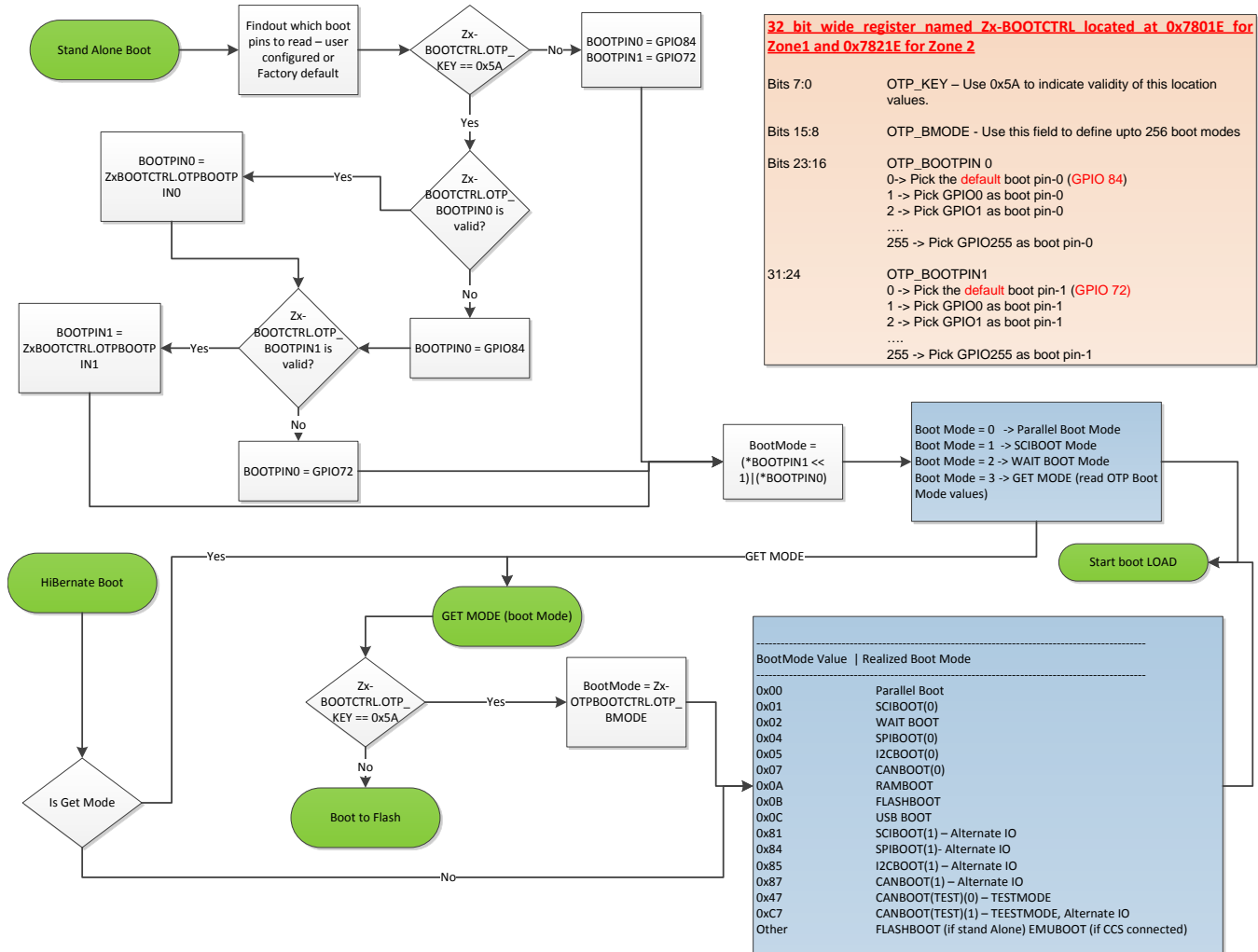
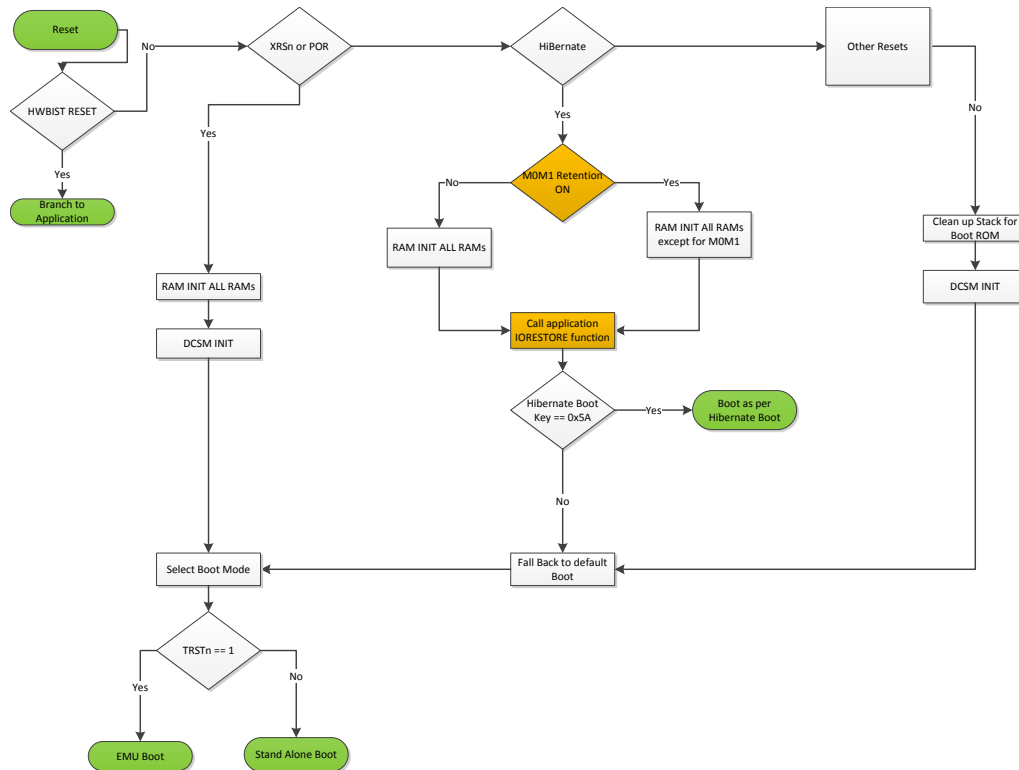
Figure 3-3. CPU1 Part 2


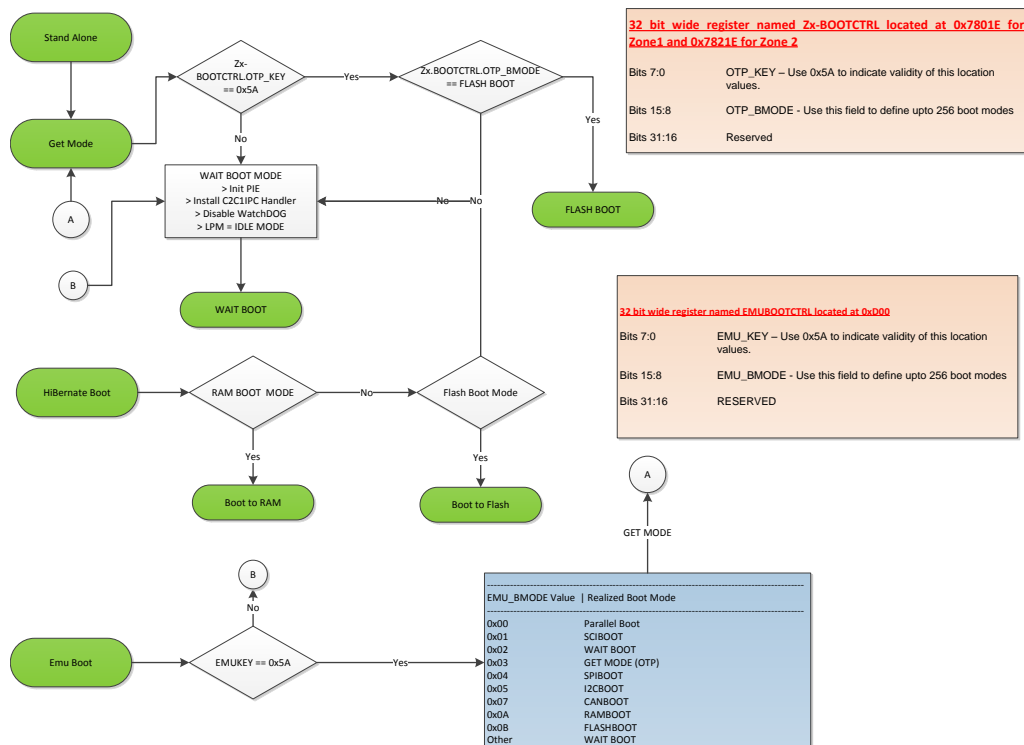
Figure 3-4. CPU1 Part 3


3.13 CPU2 Boot ROM Flow Chart

Figure 3-5. CPU2 Flow Chart (part 1)



CPU2 Flow Chart (part 2)



3.14 Boot ROM Reset Causes and Handling

Boot ROM checks the cause of reset every time device is reset and takes appropriate action as needed.

Table 3-12 shows the reset causes on this device.

Table 3-12. Reset Causes

Reset Source	CPU1 Core Reset	CPU1 Subsystem Reset	CPU2 Held Under Reset	CPU2 Core Reset	CPU2 Subsystem Reset	IO Buffers in Reset State	CPU1 Boot ROM Action	CPU2 Boot ROM Action
1 POR	Yes	Yes	Yes	Yes	Yes	yes	Default boot flow, change CLK dividers to /1 and RAM-INIT	Default boot flow
2 XRSn	Yes	Yes	Yes	Yes	Yes	Yes	Default boot flow, change CLK dividers to /1 and RAM-INIT	Default boot flow
3. CPU1.WDRSn	Yes	Yes	Yes	Yes	Yes	Yes	Default boot flow, change CLK dividers to /1 and RAM-INIT	Default boot flow
4 CPU1.NMIWDRSn	Yes	Yes	Yes	Yes	Yes	Yes	Default boot flow, change CLK dividers to /1 and RAM-INIT	Default boot flow
12 CPU2.NMIWDRSn	No	No	No	Yes	Yes	No	Exception handled by CPU1	Default boot flow
13 CPU2.WDRSn	No	No	No	Yes	Yes	No	Exception handled by CPU1	Default boot flow

3.15 Exceptions and Interrupts handling

Table 3-13 defines boot ROM action in case these exceptions occur during boot. The philosophy on CPU1 Boot ROM is to try and start the application and on CPU2 Boot ROM is to send an IPC to CPU1 and wait in loop.

Table 3-13. Exceptions Handling by Boot ROM

Exception source/type	Description	CPU1 Boot ROM Action	CPU2 Boot ROM Action
1 EFUSE-ERR Single bit	Single bit error in FUSEERR	Ignore and continue to boot	Ignore continue to boot
2 EFUSE-ERR Double bit	Not a single bit error in FUSEERR	Use CPU1.WD to reset the device (use counter overflow instead of BADKEY) – ERRORSTS pin set	DON'T CARE for CPU2
3 CLOCKFAIL	System will switch to INTOSC1	Clear the NMI, log the error and continue to boot	Clear, log and continue
4 RAMUNCERR	Double bit ECC error from RAM	Reset the device....because this error should not happen during boot as we took care of RAM-INITs	Send IPC to CPU1 and clear NMIFLG and wait forever.
5 FLUNCERR	Double bit error on FLASH	Reset the device....because this error should not happen during boot	Send IPC to CPU1 and clear NMIFLG and wait forever.
8 PIEVECTERR		Ignore, log and continue to boot	Ignore, log and continue to boot
10 RLNMI		Ignore, log and continue to boot	Send IPC to CPU1 and clear NMIFLG and continue boot.
11 CPU2WDRSn		Ignore, log and continue to boot	XXXX
12 CPU2NMIWDRSn		Ignore, log and continue to boot	XXXX
13 Unsupported PIE interrupts		Ignore and continue to boot	Send IPC to CPU1 and clear NMIFLG and continue to boot.

NOTE: Above NMI ERRORS will be logged into a RAM variable for application to read it when it starts. The location and format of this RAM variable is explained below.

3.16 CPU1 OTP Boot Configure Word

The below location is used in TI OTP to configure the behavior of boot ROM at production flow in the factory.

```
// bit 1,0 = 01 means enable PLL, else don't
// bit 7:2 = PLL divider to use - default will be all 1's, but PLL wont be used when default
// bit 9,8 = 01 means, ENABLE IPC in WAIT boot mode, else nope.
// bits 31:24 = 0x5A means the WORD is good/valid else not valid or not good
```

```
#define OTP_BOOT_CONFIGURE_WORD (Uint32)(*(volatile Uint32 *) (Uint32)(0x703EE))
```

3.17 Boot ROM Status information

3.17.1 Boot ROM Health and Status

For lack of better location CPU1 Boot ROM stores the boot/health status in a RAM location in the EBSS section allocated for boot ROM.

C1BROM STATUS - at location
0x0000002C.

CPU1 application/user can look at this location during debug or at the start of the application to check what went on during the device boot up.

The below values can be set in C1BROM STATUS[31:80.

```
#define C1_BOOTROM_START_BOOT 0x00000100 //set during the initialization
phase of bootROM
#define C1_BOOTROM_IN_FLASH_BOOT 0x00000200 //set if boot to Flash is
detected
#define C1_BOOTROM_DCSM_INIT_DONE 0x00000400 //set after DCSM is initialized
#define C1_BOOTROM_RESC_HANDLED 0x00000800 //set after all RESC is handled
#define C1_BOOTROM_HANDLED_HIBRESET 0x00001000 //set if HIB reset is detected

#define C1_BOOTROM_HANDLED_XRSN 0x00004000 //set if XRSn is detected
#define C1_BOOTROM_HANDLED_POR 0x00008000 //set if POR is serviced
#define C1_BOOTROM_IN_AN_ITRAP 0x00020000 //set if C2-
BootROM detects an iTRAP
#define C1_BOOTROM_IN_C1TOC2_BRANCH 0x00040000 //INVALID - should never be set
#define C1_BOOTROM_GOT_A_PIEMISMATCH 0x00080000 //set if pie mismatch handler is
called
#define C1_BOOTROM_GOT_A_OVF_NMI 0x00100000 //set if ovf nmi is detected
#define C1_BOOTROM_GOT_A_CPU2NMIWDRST_NMI 0x00200000 //set if cpu2nmiwdrst nmi is
detected
#define C1_BOOTROM_GOT_A_CPU2WDRST_NMI 0x00400000 //set if cpu2wdrst nmi is detected
#define C1_BOOTROM_GOT_A_RL_NMI 0x00800000 //set if RL NMI is detected
#define C1_BOOTROM_GOT_A_SYSDBG_NMI 0x01000000 //set if sysdbg nmi is detected
#define C1_BOOTROM_GOT_A_PIEVECTERR_NMI 0x02000000 //set if pievect err nmi is
detected
#define C1_BOOTROM_GOT_A_C2BISTERR_NMI 0x04000000 //set if c2-
bist error nmi is detected
#define C1_BOOTROM_GOT_A_C1BISTERR_NMI 0x08000000 //set if c1-
bist error nmi is detected
#define C1_BOOTROM_GOT_A_FLASH_UNCERR_NMI 0x10000000 //set if Flash uncerr nmi is
detected
#define C1_BOOTROM_GOT_A_RAM_UNCERR_NMI 0x20000000 //set if ram uncerr nmi is
detected
#define C1_BOOTROM_GOT_A_MCLK_NMI 0x40000000 //set if missing clock nmi is
detected
#define C1_BOOTROM_BOOT_COMPLETE 0x80000000 //set if C1-
```

BootROM is starting an application after boot

3.17.2 CPU1 Boot ROM IPC NAK Status

If CPU1 Boot ROM IPC command support is enabled, then below are the NAK status that are possibly returned by CPU1 Boot ROM in C2TOC1IPCDATAR[11:0] register. At the time of a NAK, C2TOC1IPCDATAR[31:12] will contain the C1 BOOT ROM BOOT STATUS information as shown above.

```
#define C1BROM_NAK_STATUS_INVALID_VALUE          0x00000000 //invalid value or value
when not set
#define C1BROM_NAK_STATUS_CMD_NOT_SUPPORTED      0x00000001 //NAK STATUS command not
supported
#define C1BROM_NAK_STATUS_CMD_NOT_SET_PROPERLY   0x00000002 //NAK Status command not set
properly , for ex: IPC0 set but IPC31 not set
#define C1BROM_NAK_STATUS_ALREADY_BUSY_WITH_YOUR_CMD 0x00000003 //NAK status when cpu is
trying to send a second command before first one could complete
#define C1BROM_NAK_STATUS_CMD_RESULTED_IN_ERROR 0x00000004 //NAK status command
execution resulted in an error
#define C1BROM_NAK_STATUS_CMD_CANNOT_BE_EXECUTED_NOW 0x00000005 //NAK status -
command cannot be executed now in the current state of bootROM
```

3.17.3 CPU2 Boot ROM Health and Status

The below will be given to you in C1TOC2IPCDATAR[31:12], in case of an IPC NAK and also always in C2TOC1BOOTSTS[31:12], whenever boot status is written.

```
#define C2_BOOTROM_START_BOOT          0x00000100 //set during the initialization
phase of C2-bootROM
#define C2_BOOTROM_IN_FLASH_BOOT       0x00000200 //set if boot to Flash is detected
#define C2_BOOTROM_DCSM_INIT_DONE      0x00000400 //set after DCSM is initialized
#define C2_BOOTROM_RESC_HANDLED        0x00000800 //set after all RESC is handled
#define C2_BOOTROM_HANDLED_HIBRESET    0x00001000 //set if HIB reset is detected

#define C2_BOOTROM_HANDLED_XRSN        0x00004000 //set if XRSn is detected
#define C2_BOOTROM_HANDLED_POR         0x00008000 //set if POR is serviced
#define C2_BOOTROM_IN_AN_ITRAP         0x00020000 //set if C2-
BootROM detects an iTRAP
#define C2_BOOTROM_IN_C1TOC2_BRANCH    0x00040000 //set if C2-
BootROM is doing a mtoc branch
#define C2_BOOTROM_GOT_A_PIEMISMATCH   0x00080000 //set if pie mismatch handler is
called
#define C2_BOOTROM_GOT_A_OVF_NMI       0x00100000 //set if ovf nmi is detected
#define C2_BOOTROM_GOT_A_CPU2NMIWDRST_NMI 0x00200000 // - should never be set
#define C2_BOOTROM_GOT_A_CPU2WDRST_NMI 0x00400000 // - should never be set
#define C2_BOOTROM_GOT_A_RL_NMI        0x00800000 //set if RL NMI is detected
#define C2_BOOTROM_GOT_A_SYSDBG_NMI    0x01000000 //set if sysdbg nmi is detected
#define C2_BOOTROM_GOT_A_PIEVECTERR_NMI 0x02000000 //set if pievect err nmi is
detected
#define C2_BOOTROM_GOT_A_C2BISTERR_NMI 0x04000000 //set if c2-
bist error nmi is detected
#define C2_BOOTROM_GOT_A_C1BISTERR_NMI 0x08000000 //set if c1-
bist error nmi is detected
#define C2_BOOTROM_GOT_A_FLASH_UNCERR_NMI 0x10000000 //set if Flash uncerr nmi is
detected
#define C2_BOOTROM_GOT_A_RAM_UNCERR_NMI 0x20000000 //set if ram uncerr nmi is
detected
#define C2_BOOTROM_GOT_A_MCLK_NMI      0x40000000 //set if missing clock nmi is
detected
#define C2_BOOTROM_BOOT_COMPLETE       0x80000000 //set if C2-
BootROM is starting an application after boot
```

C2TOC1BOOTSTS[11:0] – these bits will carry the below status information.

```
#define C2_BOOTROM_BOOTSTS_CTOM_IGNORE 0x00000000 //invalid
status - tells that CPU2 has not filled in a valid value yet
#define C2_BOOTROM_BOOTSTS_SYSTEM_START_BOOT 0x00000001 //tells that
CPU2 has started to boot, but not completed the boot process yet
```

```
#define C2_BOOTROM_BOOTSTS_SYSTEM_READY 0x00000002 //tells that
CPU2 has completed the boot and is ready for C1TOC2 IPC commands
#define C2_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_ACK 0x00000003 //tells that
CPU2 ACKs the command in C1TOC2BOOTMODE register
#define C2_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_NAK_STATUS_NOT_SUPPORTED 0x00000004 //tells CPU2
doesn't support the command in C1TOC2BOOTMODE register
#define C2_BOOTROM_BOOTSTS_CTOM_BOOT_CMD_NAK_STATUS_BUSY_WITH_BOOT 0x00000005 //tells that
CPU2 NAKs the current boot command in C1TOC2BOOTMODE register
```

3.17.4 CPU2 Boot ROM IPC NAK status

C1TOC2IPCDATAR[11:0] register values, when CPU2 BootROM system NAK's an IPC Command sent by CPU1 application.

C1TOC2IPCDATAR[31:12] register value stores the C2-BootROM health status as explained in above section

```
#define C2_BOOTROM_NAK_STATUS_INVALID_VALUE 0x00000000 //invalid value or value
when not set
#define C2_BOOTROM_NAK_STATUS_CMD_NOT_SUPPORTED 0x00000001 //NAK STATUS command not
supported
#define C2_BOOTROM_NAK_STATUS_CMD_NOT_SET_PROPERLY 0x00000002 //NAK Status command not
set properly , for ex: IPC0 set but IPC31 not set
#define C2_BOOTROM_NAK_STATUS_ALREADY_BUSY_WITH_YOUR_CMD 0x00000003 //NAK status when master
is trying to send a second command before first one could complete
#define C2_BOOTROM_NAK_STATUS_CMD_RESULTED_IN_ERROR 0x00000004 //NAK status command
execution resulted in an error
#define C2_BOOTROM_NAK_STATUS_CMD_CANNOT_BE_EXECUTED_NOW 0x00000005 //NAK status -
command cannot be executed now in the current state of bootROM
```

3.18 Boot and IPC Commands

3.18.1 BOOTMODE Commands

This section explains the boot mode commands used between CPU1 and CPU2. BootROM running on CPU2 core will decode the boot command from CPU1 system and take appropriate action.

3.18.1.1 C1TOC2BOOTMODE Commands

CPU1 Boot ROM does not communicate with CPU2 Boot ROM, it is the user application running on CPU1 which will talk to CPU2 Boot ROM using C1TOC2BOOTMODE commands. Please refer to the C1TOC2IPC Commands table on the supported IPC boot mode commands by CPU2 Boot ROM.

Before sending the boot mode IPC command to CPU2 Boot ROM, the CPU1 application is supposed to do the needed GPIO mux configurations for the peripheral IO pins and assign the peripheral to CPU2 by appropriately configuring the CPUSEL register.

CPU2 peripheral loaders doesn't configure any of the GPIO mux options but simply configure the peripheral as needed for the application load. The data protocols followed by CPU2 on peripheral loaders are same as in CPU1.

3.18.2 CPU1 Boot ROM Supported IPC Commands

CPU1 Boot ROM supports the following IPC commands. The feature can be used by CPU2 applications to command CPU1 Boot ROM to configure peripherals or IOs. CPU1 Boot ROM in WAIT BOOT MODE supports the below IPC commands. Note that WATCHDOG on CPU1 is disabled in this mode.

3.18.2.1 C2TOC1IPC Commands Table as Decoded by CPU1 Boot ROM

Table 3-14. C2TOC1IPC Commands Table ⁽¹⁾

Value	IPCRECVCOM (CPU2 - R/W, CPU1- R)	IPCRECVADDR (CPU2 - R/W, CPU1- R)	IPCRECVDATA (CPU2 - R/W, CPU1- R)	IPCLOCALREPLY (CPU2 - R, CPU1 - R/W)	C2TOC1IPCFLG[31] = ? C2TOC1IPCFLAG[0] = 0	Description
0	C2C1_BROM_IPC_COMMAND_ILLEGAL	_NOT-USED_	_NOT-USED_	Look at error codes table	0x01	Illegal command
1	C2C1_BROM_IPC_SET_BITS_16	Address of the 16 bit register	Data in C2TOC1IPCDATAW[15:0]	Data read back from address after write	0x00 = Command success	*(address) = data;
2	C2C1_BROM_IPC_SET_BITS_32	Address of the 32 bit register	Data;	Data read back from address after write	Same as above	*(address) = data;
3	C2C1_BROM_IPC_CLEAR_BIT_S_16	Address of the 16 bit register	Data in C2TOC1IPCDATAW[15:0]	Data read back after write	Same as above	*(address) &= ~data;
4	C2C1_BROM_IPC_CLEAR_BIT_S_32	Address of the 32 bit register	Data	Data read back after write	Same as above	*(address) &= ~data;
5	C2C1_BROM_IPC_DATA_WRITE_16	Address of the 16 bit register	Data in MTOCIPCDATAW[15:0]	Data read back from the address	Same as above	*(address) = data;
6	C2C1_BROM_IPC_DATA_WRITE_32	Address of the 32 bit register	Data	Same as above	Same as above	*(address) = data;
7	C2C1_BROM_IPC_DATA_READ_16	Address of the 16 bit register	--NOT_USED--	Data in C2TOC1IPCDATAR[15:0]	Same as above	C2TOC1IPCDATAR[15:0] = *(address); Only 16 bit read from address
8	C2C1_BROM_IPC_DATA_READ_32	Address of the 32 bit register	Same as above	32 bit data	Same as above	C2TOC1IPCDATAR[31:0] = *(address); 32 bits read from address
9	C2C1_BROM_IPC_SET_BITS_PROTECTED_16	Address of the 16 bit register	Data in C2TOC1IPCDATAW[15:0]	Data read back from address after write	0x00 = Command success	EALLOW; *(address) = data; EDIS;
10	C2C1_BROM_IPC_SET_BITS_PROTECTED_32	Address of the 32 bit register	Data;	Data read back from address after write	Same as above	EALLOW; *(address) = data; EDIS;
11	C2C1_BROM_IPC_CLEAR_BIT_S_PROTECTED_16	Address of the 16 bit register	Data in C2TOC1IPCDATAW[15:0]	Data read back after write	Same as above	EALLOW; *(address) &= ~data; EDIS;
12	C2C1_BROM_IPC_CLEAR_BIT_S_PROTECTED_32	Address of the 32 bit register	Data	Data read back after write	Same as above	EALLOW; *(address) &= ~data; EDIS;
13	C2C1_BROM_IPC_DATA_WRITE_PROTECTED_16	Address of the 16 bit register	Data in C2TOC1IPCDATAW[15:0]	Data read back from the address	Same as above	EALLOW; *(address) &= ~data; EDIS;
14	C2C1_BROM_IPC_DATA_WRITE_PROTECTED_32	Address of the 32 bit register	Data	Same as above	Same as above	EALLOW; *(address) &= ~data; EDIS;
15	C2C1_BROM_IPC_DATA_READ_PROTECTED_16	Address of the 32 bit register	--NOT_USED--	Data in C2TOC1IPCDATAR[15:0]	Same as above	EALLOW; C2TOC1IPCDATAR[15:0] = *(address); EDIS; Only 16 bit read from address
16	C2C1_BROM_IPC_DATA_READ_PROTECTED_32	Address of the 32 bit register	Data	Same as above	Same as above	EALLOW; C2TOC1IPCDATAR[31:0] = *(address); EDIS; 32 bits read from address

⁽¹⁾ All 32-bit operations are done in little endian format (C28X is 16 bit addressable).

Example: a 32-bit IPC write is handles as below:

- Data[15:0] is written in address
- Data [31:16] is written in address+1

3.18.3 CPU2 Boot ROM Supported IPC Commands

3.18.3.1 C1TOC2IPC Commands Table as Decoded by CPU2 Boot ROM

Table 3-15. C1TOC2IPC Commands Table

Value	IPCRCVCOM (CPU1 - R/W, CPU2 R)	IPCRCVADDR (CPU1 - R/W, BCPU2- R)	IPCRCVDATA (CPU1 - R/W, CPU2- R)	IPCLOCALREPL Y (CPU1 - R, CPU2 - R/W)	C1TOC2IPCFLG [31] = ? C1TOC2IPCFLA G[0] = 0	Description	
0	C1C2_BROM_IPC_COMM AND_ILLEGAL	_NOT-USED_	_NOT-USED_	Look at error codes table	0x01	Illegal command	
1.	C1C2_BROM_IPC_SET_BI TS_16	Address of the 16 bit register	Data in C1TOC2IPCDATA W[15:0]	Data read back from address after write	0x00 = Command success	*(address) = data;	
2.	C1C2_BROM_IPC_SET_BI TS_32	Address of the 32 bit register	Data;	Data read back from address after write	Same as above	*(address) = data;	
3.	C1C2_BROM_IPC_CLEAR _BITS_16	Address of the 16 bit register	Data in C1TOC2IPCDATA W[15:0]	Data read back after write	Same as above	*(address) &= ~data;	
4.	C1C2_BROM_IPC_CLEAR _BITS_32	Address of the 32 bit register	Data	Data read back after write	Same as above	*(address) &= ~data;	
5.	C1C2_BROM_IPC_DATA_ WRITE_16	Address of the 16 bit register	Data in MTOCIPCDATAW [15:0]	Data read back from the address	Same as above	*(address) = data;	
6.	C1C2_BROM_IPC_DATA_ WRITE_32	Address of the 32 bit register	Data	Same as above	Same as above	*(address) = data;	
7	C1C2_BROM_IPC_DATA_ READ_16	Address of the 16 bit register	--NOT_USED--	Data in C1TOC2IPCDATA R[15:0]	Same as above	C1TOC2IPCDATAR [15:0] = *(address); Only 16 bit read from address	
8	C1C2_BROM_IPC_DATA_ READ_32	Address of the 32 bit register	Same as above	32 bit data	Same as above	C1TOC2IPCDATAR [31:0] = *(address); 32 bits read from address	
9	C1C2_BROM_IPC_SET_BI TS_PROTECTED_16	Address of the 16 bit register	Data in C1TOC2IPCDATA W[15:0]	Data read back from address after write	0x00 = Command success	EALLOW; *(address) = data; EDIS;	
10	C1C2_BROM_IPC_SET_BI TS_PROTECTED_32	Address of the 32 bit register	Data;	Data read back from address after write	Same as above	EALLOW; *(address) = data; EDIS;	
11	C1C2_BROM_IPC_CLEAR _BITS_PROTECTED_16	Address of the 16 bit register	Data in C1TOC2IPCDATA W[15:0]	Data read back after write	Same as above	EALLOW; *(address) &= ~data; EDIS;	
12	C1C2_BROM_IPC_CLEAR _BITS_PROTECTED_32	Address of the 32 bit register	Data	Data read back after write	Same as above	EALLOW; *(address) &= ~data; EDIS;	
13	C1C2_BROM_IPC_DATA_ WRITE_PROTECTED_16	Address of the 16 bit register	Data in C1TOC2IPCDATA W[15:0]	Data read back from the address	Same as above	EALLOW; *(address) = data; EDIS;	
14	C1C2_BROM_IPC_DATA_ WRITE_PROTECTED_32	Address of the 32 bit register	Data	Same as above	Same as above	EALLOW; *(address) = data; EDIS;	
15	C1C2_BROM_IPC_DATA_ READ_PROTECTED_16	Address of the 16 bit register	--NOT_USED--	Data in C1TOC2IPCDATA R[15:0]	Same as above	EALLOW; C1TOC2IPCDATAR [15:0] = *(address); Only 16 bit read from address	
16	C1C2_BROM_IPC_DATA_ READ_PROTECTED_32	Address of the 32 bit register	Same as above	32 bit data	Same as above	EALLOW; C1TOC2IPCDATAR [31:0] = *(address); EDIS; 32 bits read from address	

Table 3-15. C1TOC2IPC Commands Table (continued)

Value	IPCRCVCOM (CPU1 - R/W, CPU2 R)	IPCRCVADDR (CPU1 - R/W, BCPU2- R)	IPCRCVDATA (CPU1 - R/W, CPU2- R)	IPCLOCALREPL Y (CPU1 - R, CPU2 - R/W)	C1TOC2IPCFLG [31] = ? C1TOC2IPCFLA G[0] = 0	Description	
17	C1C2_BROM_IPC_BRANC H_CALL	Address where to branch to	_NOT- USED_BY_BOOT ROM (code at the branch can use this though)	NOT- USED_BY_BOOT ROM (code at the branch can use this though)	Same as above	C2-BootROM will jump to the address in ADDR register and starts executing the code from that address. PIE will be enabled when this branch occurs, it is upto the application to disable and reload PIE interrupt handlers if it wants to.	
18	C1C2_BROM_IPC_FUNC TION_CALL	Address of the function	Parameter for the function call	Result of function call (return value if any from function)	Same as above	C2-BootROM will jump to the address in ADDR register and starts executing the code from that address; Data in DATAW register can be used as parameter to the function call. C- BootROM returns back to where it was after servicing the function call. Function call is performed from inside the interrupt service routine on Aria so user has to keep this in mind.	
19	C1C2_BROM_IPC_EXECU TE_ BOOTMODE_CMD <ul style="list-style-type: none"> C1TOC2BOOTMO DE = 0xA, C1C2_BROM_BO OTMODE_ BOOT_FROM_RA M C1TOC2BOOTMO DE = 0xB, C1C2_BROM_BO OTMODE_ BOOT_FROM_FLA SH C1TOC2BOOTMO DE = 0x1, C1C2_BROM_BO OTMODE_ BOOT_FROM_SCI C1TOC2BOOTMO DE = 0x4, C1C2_BROM_BO OTMODE_ BOOT_FROM_SPI 	--NOT_USED--	--NOT_USED--	--NOT_USED--	Same as above	Execute loaders as per requested value in C1TOC2BOOTMO DE register.	

Table 3-15. C1TOC2IPC Commands Table (continued)

Value	IPCRCVCOM (CPU1 - R/W, CPU2 R)	IPCRCVADDR (CPU1 - R/W, BCPU2- R)	IPCRCVDATA (CPU1 - R/W, CPU2- R)	IPCLOCALREPL Y (CPU1 - R, CPU2 - R/W)	C1TOC2IPCFLG [31] = ? C1TOC2IPCFLA G[0] = 0	Description	
	<ul style="list-style-type: none"> C1TOC2BOOTMODE = 0x5, C1C2_BROM_BOOTMODE_ BOOT_FROM_I2C C1TOC2BOOTMODE = 0x0, C1C2_BROM_BOOTMODE_ BOOT_FROM_PARALLEL C1TOC2BOOTMODE = 0x7, C1C2_BROM_BOOTMODE_ BOOT_FROM_CANN 						

3.18.3.2 CPU2 Boot ROM Sent IPC Commands in Case of Errors

CPU2 Boot ROM sends the below IPC commands to CPU1 in case there is an error. If CPU1 application enables C2C1IPCINT1 while CPU2 Boot ROM is running then the C2TOC1IPC COM register values upon receiving an IPC interrupt are as defined below:

```
#define C2_BOOTROM_IPC_CTOM_COMMAND_ILLEGAL          0x00000000    //invalid command value -
//default value when starting boot
#define C2_BOOTROM_IPC_CTOM_CONTROL_SYSTEM_IN_ITRAP  0xFFFFFFFFE    //message tells that CPU2
//has got an iTRAP, address where iTrap occurred will be in IPC ADDR register
#define C2_BOOTROM_IPC_CTOM_PIE_INTERRUPT_NOT_SUPPORTED 0xFFFFFFFFD    //message tells that CPU2
//got a spurious PIE interrupt, intr no. will be in IPCDATAW register
#define C2_BOOTROM_IPC_CTOM_PIE_VECTOR_ADDRESS_MISMATCH 0xFFFFFFFFC    //message tells that CPU2
//got a PIE vector mismatch error
#define C2_BOOTROM_IPC_CTOM_CONTROL_SYSTEM_IN_FLUNCERR 0xFFFFFFFFB    //message tells that CPU2
//got a Flash uncorrectable error
#define C2_BOOTROM_IPC_CTOM_CONTROL_SYSTEM_IN_RAMUNCERR 0xFFFFFFFFA    //message tells that CPU2
//got a RAM uncorrectable error
```

3.19 Device Boot Process Timeline Diagram

Table 3-16 gives an overview of synchronization between CPU1 and CPU2 boot ROMs.

Table 3-16. Time Diagram

Time Line	CPU1	CPU2
T0	POR indicates that device is not yet fully powered and is in reset	POR indicates that device is not yet fully powered and is in reset

Table 3-16. Time Diagram (continued)

Time Line	CPU1	CPU2
T1	CPU1 System Configuration: <ul style="list-style-type: none"> • RESC cause check • PLL POWER UP • FALSH POWER UP • RAM INITS • DCSM INIT • NMI ENABLE • PLL BYPASS 	<ul style="list-style-type: none"> • CPU2 is held in reset
T2	Bring CPU1 out of reset	CPU2 System Config: <ul style="list-style-type: none"> • RESC check • RAM INIT • DCSM INIT • NMI ENABLE
T3	EMU BOOT or STANDALONE BOOT or HIB BOOT <ul style="list-style-type: none"> • If WAIT BOOT go to wait boot loop • If boot from peripheral download application code from peripheral and save it in RAM and jump to start of application. • If WAIT BOOT go to wait boot loop 	HIBRESC check READ BOOT MODE BOOT TO RAM/BOOT TO FLASH Init PIE and WAIT BOOT MODE

3.20 Boot ROM GPIO Configurations:

Default boot mode GPIO pins:

Boot mode pin 0 : GPIO 84

Boot mode pin 1 : GPIO 72

The boot mode pins are configurable by user by programming proper OTPKEY.

SCIBoot – IO Option 1

SCITX: GPIO84

SCIRX: GPIO85

SCIBoot – IO Option 2

SCITX: GPIO28

SCIRX: GPIO29

Bit parallel bootloader GPIO pins

D0 - GPIO65

D1 - GPIO64

D2 - GPIO58

D3 - GPIO59

D4 - GPIO60

D5 - GPIO61

D6 - GPIO62

D7 - GPIO63

HOST_CTRL - GPIO70

DSP_CTRL - GPIO69

CAN Boot – IO Option 1

CAN0_RX - GPIO70

CAN0_TX - GPIO71

CAN Boot – IO Option 2

CAN0_RX - GPIO62
CAN0_TX - GPIO63

I2C Boot - IO Option 1

I2C0_SDAA - GPIO91
I2C0_SCLA - GPIO92

I2C Boot - IO Option 2

I2C0_SDAA - GPIO32
I2C0_SCLA - GPIO33

SPI Boot – IO Option 1

SPISIMOA - GPIO58
SPISOMIA - GPIO59
SPICLKA - GPIO60
SPISTEA - GPIO61

SPI Boot – IO option 2

SPISIMOA - GPIO16
SPISOMIA - GPIO17
SPICLKA - GPIO18
SPISTEA - GPIO19

USB Boot

USB0DM - GPIO42
USB0DP - GPIO43
(the mode implemented is USB device type, Device Firmware Upgrade Class)

3.21 Boot ROM Memory Map

The memory map of the Boot ROM is shown in [Section 3.21.1 Figure 3-6](#).

Figure 3-6. Boot ROM Memory Map

ENTRY POINT SYMBOL: "_InitBoot" address: 003ff16a

MEMORY CONFIGURATION

name	origin	length	used	unused	attr	fill
-----	-----	-----	-----	-----	----	-----
--						
PAGE 0:						
SYSBIOS_FLASH	00082000	00002000	00000824	000017dc	RWIX	
Z1_SCC_ROM	003f0000	00000800	00000800	00000000	RWIX	ffff
Z2_SCC_ROM	003f0800	00000800	00000800	00000000	RWIX	ffff
TI_RESERVED	003f1000	00000008	00000008	00000000	RWIX	ffff
Z1_SECURE_ROM	003f1008	00006ff8	00006ff8	00000000	RWIX	ffff
ROM_SIGNATURE	003f8000	00000002	00000002	00000000	RWIX	ffff
SYSBIOS_ROM	003f8002	00001e0e	00001e0e	00000000	RWIX	ffff
PLCTABLES	003f9e10	00003a08	00003a08	00000000	RWIX	ffff
PLCTABLES2	003fd818	00000600	00000600	00000000	RWIX	ffff
BOOT	003fde18	0000211a	0000211a	00000000	RWIX	ffff
CRCTABLE_ROM	003fff32	00000008	00000008	00000000	RWIX	ffff
BIST_SIGNATURE	003fff3a	00000040	00000040	00000000	RWIX	
VERSION	003fff7a	00000002	00000002	00000000	RWIX	
CHECKSUM	003fff7c	00000042	00000042	00000000	RWIX	
VECS	003fffbе	00000042	00000042	00000000	RWIX	

3.21.1 F2837x Memory Map – CPU2

Figure 3-7. F2837x Memory Map – CPU2

```

*****
TMS320C2000 Linker PC v6.1.3
*****
>> Linked Wed Apr 10 21:23:14 2013

OUTPUT FILE NAME:  <F2837x_cpu02_bootROM.out>
ENTRY POINT SYMBOL: "_InitBoot"  address: 003fe649

MEMORY CONFIGURATION

      name          origin    length    used    unused    attr    fill
-----
PAGE 0:
SYSBIOS_FLASH      00080010  00001ff0  000007be  00001832  RWIX
Z1_SCC_ROM          003f0000  00000800  00000800  00000000  RWIX  ffff
Z2_SCC_ROM          003f0800  00000800  00000800  00000000  RWIX  ffff
TI_RESERVED         003f1000  00000008  00000008  00000000  RWIX  ffff
Z1_SECURE_ROM       003f1008  00006ff8  00006ff8  00000000  RWIX  ffff
ROM_SIGNATURE       003f8000  00000002  00000002  00000000  RWIX  ffff
SYSBIOS_ROM         003f8002  00001e0e  00001e0e  00000000  RWIX
RESERVED            003f9e10  00003a08  00003a08  00000000  RWIX  ffff
BOOT                003fd818  0000271a  0000271a  00000000  RWIX  ffff
CRCTABLE_ROM        003fff32  00000008  00000008  00000000  RWIX  ffff
BIST_SIGNATURE       003fff3a  00000040  00000040  00000000  RWIX
VERSION             003fff7a  00000002  00000002  00000000  RWIX
CHECKSUM            003fff7c  00000042  00000042  00000000  RWIX
VECS                003fffbе  00000042  00000042  00000000  RWIX

```

3.22 CPU1 and CPU2 ROM REVISION Information

ROM revision and release date information is stored 32 bits from the location 0x003FFF7A as shown below:

- CPU1 ROM REVISION is as below starting from 0x003FFF7A location

```

.sect ".Version"
.global  _c1brom_version
c1brom_version:
.word 0x0100      ; Boot ROM Version v0.1
.word 0x0413      ; Month/Year: (ex: 0x0109 = 1/09 = Jan 2009)

```

- CPU2 ROM REVISION is as below starting from 0x003FFF7A location

```

.sect ".Version"
.global  _c2brom_version
c2brom_version:
.word 0x0100      ; Boot ROM Version v0.1
.word 0x0413      ; Month/Year: (ex: 0x0109 = 1/09 = Jan 2009)

```

In the 16-bit REVISION word at location 0x3FFF7A, LSB 8 bits need to be reserved to store the Secure ROM contents revision. Since Secure ROM is EXEONLY boot ROM revision locations can be read by application to find out the revision of code in secure ROM.

On REV0 of the Silicon there are no Secure ROM libraries, so the revision is 0x0 for this on both CPU1 and CPU2.

3.23 RAM and Flash Usage and Application Entry Points

Below is the RAM memory usage on CPU1 Boot ROM. All the stack memory will be Zeroed out by boot ROM whenever boot ROM is run from start.

The SYSBIOS_RAM is a special section which is used by SYSBIOS in boot ROM. This memory must be reserved by users application and should not be used by application if they are planning on using SYSBIOS is ROM.

```
EBSS          : origin = 0x002,    length = 0x000040
STACK         : origin = 0x042,    length = 0x0000E0
SYSBIOS_RAM   : origin = 0x780,    length = 0x000080
```

3.23.1 Reserved RAM Memory for CPU2-Boot ROM

Below is the RAM memory usage on CPU2 Boot ROM. All the stack memory will be Zeroed out by boot ROM whenever boot ROM is run from start.

The SYSBIOS_RAM is a special section which is used by SYSBIOS in boot ROM. This memory must be reserved by users application and should not be used by application if they are planning on using SYSBIOS is ROM.

```
EBSS          : origin = 0x002,    length = 0x00001E
STACK         : origin = 0x042,    length = 0x000060
SYSBIOS_RAM   : origin = 0x780,    length = 0x000080
```

3.23.2 CPU1 and CPU2 RAM Entry Point

RAM entry point is fixed at 0x00000000, user applications when using boot-to-ram option are supposed to place their entry point which branches to main() at this location. Boot ROM when set to boot-to-RAM branches to this location.

3.23.3 CPU1 and CPU2 Flash Entry Point

Flash entry point is fixed at 0x00080000, user applications when using boot-to-flash option are supposed to place their entry point which branches to main() at this location. Boot ROM when set to boot-to-flash branches to this location.

3.23.4 CPU1/ and CPU2 Flash Reserved Memory

The Flash memory described below is reserved for SYSBIOS using SYSBIOS libraries in ROM. The locations in Flash memory will be programmed with needed SYSBIOS components linked with the application.

If user is not planning on using SYSBIOS in ROM then they are free to use this memory as needed by application. But if user is planning on using SYSBIOS in ROM then this memory is supposed to be reserved and preferably this sector is supposed to be made un-secure or made sure that no secure Zone claims this sector.

```
SYSBIOS_Flash : origin = 0x00082000, length = 0x00000824
```

3.24 ROM Wait States

ROM memory on this device is not ZERO-WAIT STATE by default. ROM memory is supports prefetch enable and disable configuration to support better execution speeds at different clock frequency configurations as shown below.

By default, ROM is 1-Wait State and pre-fetch is disabled, but if the user is executing his application below 180 MHz then Pre-fetch can be enabled and if the user application is running below 150 MHz then ROM can be configured for 0-WS operation as shown in [ROM Wait Sates](#).

Table 3-17. ROM Wait States

WAITSTATE Disable Bit (bit 0 at 0x5F540)	Pre-Fetch Enable Bit Bit 0 at address (0x5e608)	ROM Configuration
0	0	WS enabled Pre-fetch disabled Max Frequency : 200 MHz

Table 3-17. ROM Wait States (continued)

WAITSTATE Disable Bit (bit 0 at 0x5F540)	Pre-Fetch Enable Bit Bit 0 at address (0x5e608)	ROM Configuration
0	1	WS enabled Pre-fetch *enabled* Max Frequency : 180 MHz
1	x	0-wait Pre-fetch disabled Max Frequency : 150 MHz

3.25 Device Boot Modes Description

All the peripheral boot modes except USB on CPU1 support two options per peripheral and each option is configured to work on different set of RX/TX IOs. Depending on the application design user can choose either option.

Peripheral boot modes on CPU2 are supposed to be started by application running on CPU1 and this application is supposed to be configure the peripheral and needed IOs for CPU2 before sending the proper boot mode IPC command to CPU2 Boot ROM.

The boot modes below are the same as they were on the legacy C2000 device.

- SCI Boot Mode
- SPI Boot Mode
- I2C Boot Mode
- Parallel Boot Mode
- DCAN Boot Mode

The following boot mode is new on the device:

- USB Boot Mode

USB Boot Mode implemented supports Device Firmware Upgrade class (DFU) and is the same as the Flash-based USB loader.

The following sections explain each of the supported peripheral boot mode(s) on CPU1/CPU2 Boot ROM, in detail.

The data transfer protocols or stream structures that allow boot data transfer between boot ROM and Host device, are compatible to the respective bootloaders on the Piccolo class of C2000 devices. This allows the user to reuse the same tools used to send boot data to Piccolo(s) to also send boot data to boot ROM.

3.25.1 Boot 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 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 for both 8-bit and 16-bit data stream formats as the number of 16-bit words in the block. 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 make up 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 3-18. General Structure Of Source Program Data Stream In 16-Bit Mode

Word	Contents
1	10AA (KeyValue for memory width = 16bits)
2	Register initialization value or reserved for future use
3	Register initialization value or reserved for future use
4	Register initialization value or reserved for future use
5	Register initialization value or reserved for future use
6	Register initialization value or reserved for future use
7	Register initialization value or reserved for future use
8	Register initialization value or reserved for future use
9	Register initialization value or reserved for future use
10	Entry point PC[22:16]
11	Entry point PC[15:0]
12	Block size (number of words) of the first block of data to load. If the block size is 0, this indicates the end of the source program. Otherwise another section follows.
13	Destination address of first block Addr[31:16]
14	Destination address of first block Addr[15:0]
15	First word of the first block in the source being loaded
...	...
...	...
.	Last word of the first block of the source being loaded
.	Block size of the 2nd block to load.
.	Destination address of second block Addr[31:16]
.	Destination address of second block Addr[15:0]
.	First word of the second block in the source being loaded
.	...
.	Last word of the second block of the source being loaded
.	Block size of the last block to load
.	Destination address of last block Addr[31:16]
.	Destination address of last block Addr[15:0]
.	First word of the last block in the source being loaded
...	...
...	...
n	Last word of the last block of the source being loaded
n+1	Block size of 0000h - indicates end of the source program

Example 3-1. Data Stream Structure 16-bit

```

10AA                ; 0x10AA 16-bit key value
0000 0000 0000 0000 ; 8 reserved words
0000 0000 0000 0000
003F 8000           ; 0x003F8000 EntryAddr, starting point after boot load completes
0005                ; 0x0005 - First block consists of 5 16-bit words
003F 9010           ; 0x003F9010 - First block will be loaded starting at 0x3F9010
0001 0002 0003 0004 ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
0005
0002                ; 0x0002 - 2nd block consists of 2 16-bit words
003F 8000           ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
7700 7625           ; Data loaded = 0x7700 0x7625
0000                ; 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

```

In 8-bit mode, the least significant byte (LSB) of the word is sent first followed by the most significant byte (MSB). For 32-bit values, such as a destination address, the most significant word (MSW) is loaded first, followed by the least significant word (LSW). The bootloaders take this into account when loading an 8-bit data stream.

Table 3-19. 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 3-2. 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
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:

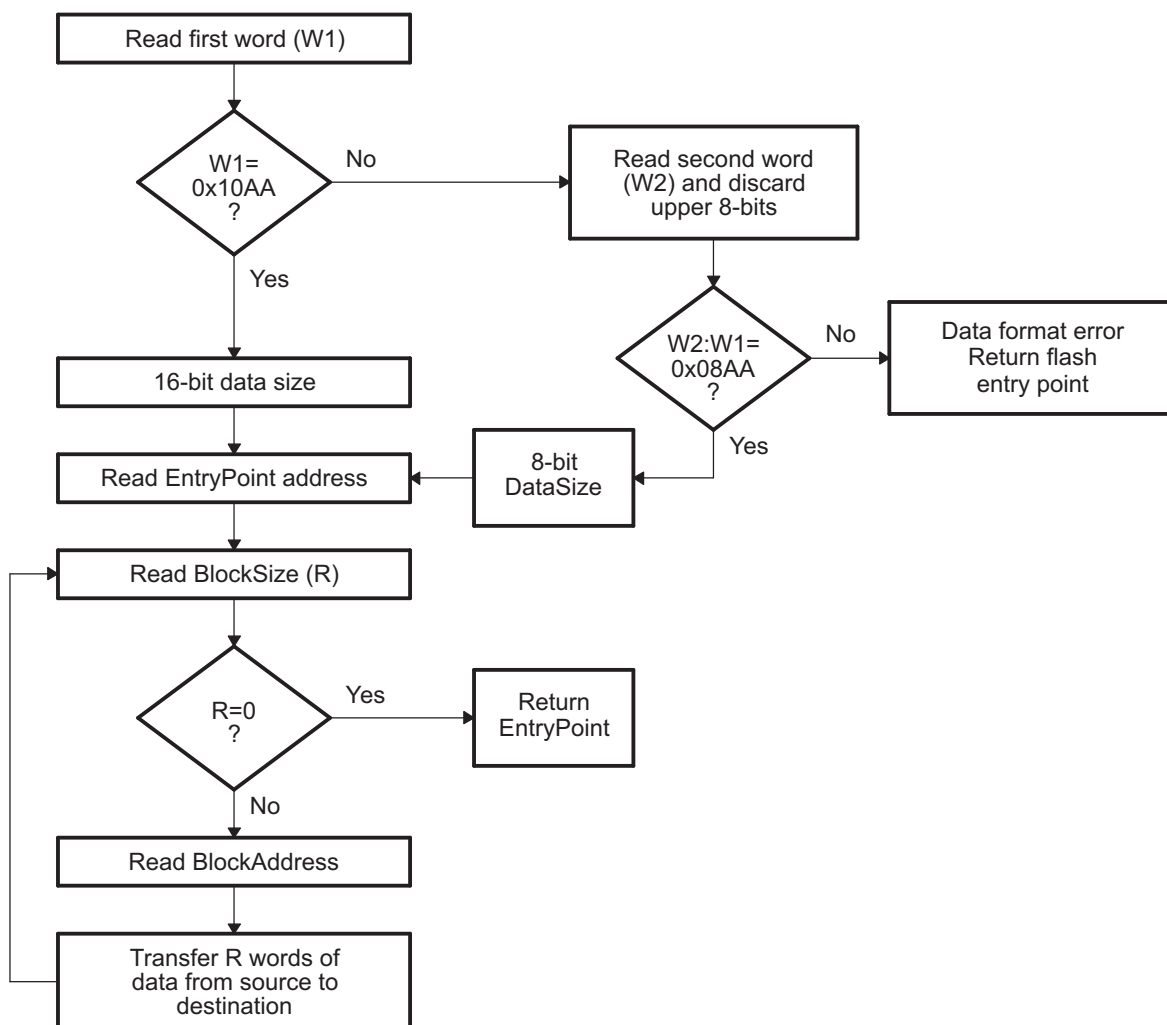
Location	Value
0x3F9010	0x0001
0x3F9011	0x0002
0x3F9012	0x0003
0x3F9013	0x0004
0x3F9014	0x0005
0x3F8000	0x7700
0x3F8001	0x7625

PC Begins execution at 0x3F8000

3.25.2 Basic Data Transfer Procedure

Figure 3-8 illustrates the basic process a bootloader uses to determine whether 8-bit or 16-bit data stream has been selected, transfer that data, and start program execution. This process occurs after the bootloader finds the valid boot mode selected by the state of $\overline{\text{TRST}}$ and GPIO pins.

The loader first compares the first value sent by the host against the 16-bit key value of 0x10AA. If the value fetched does not match then the loader will read a second value. This value will be combined with the first value to form a word. This will then be checked against the 8-bit key value of 0x08AA. If the loader finds that the header does not match either the 8-bit or 16-bit key value, or if the value is not valid for the given boot mode then the load will abort.

Figure 3-8. Bootloader Basic Transfer Procedure


8-bit and 16-bit transfers are not valid for all boot modes. If only one mode is valid, then this decision tree is skipped and the key value is only checked for correctness. See the info specific to a particular bootloader for any limitations.

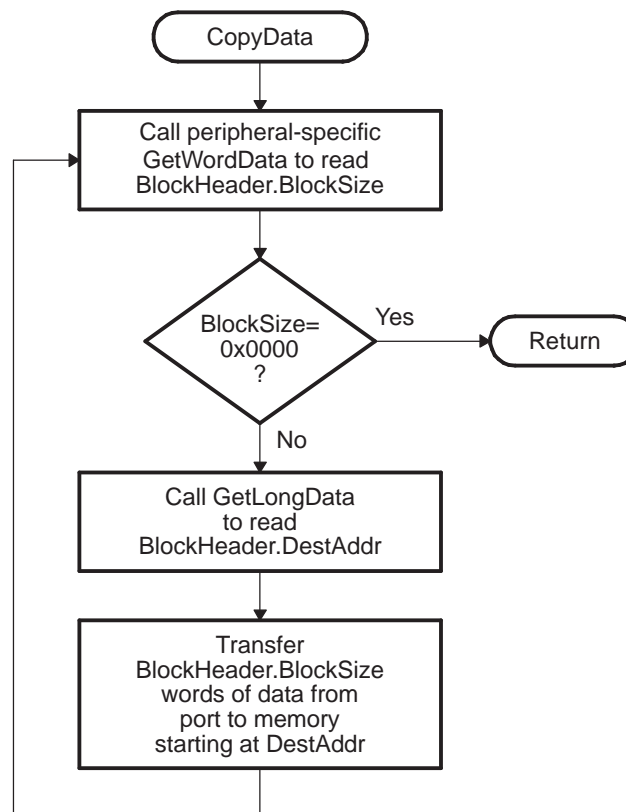
In 8-bit mode, the LSB of the 16-bit word is read first followed by the MSB.

3.25.3 CopyData Function

Each of the bootloaders uses the same function to copy data from the port to the device's SARAM. This function is the CopyData() function. This function uses a pointer to a GetWordData function that is initialized by each of the loaders to properly read data from that port. For example, when the SPI loader is evoked, the GetWordData function pointer is initialized to point to the SPI-specific SPI_GetWordData function. Thus when the CopyData() function is called, the correct port is accessed. The flow of the CopyData function is shown in Figure 3-9.

Note: BlockSize must be less than 0xFFFF for correct operation of the CopyData function. This means the max possible value of BlockSize is 0xFFFE, not 0xFFFF.

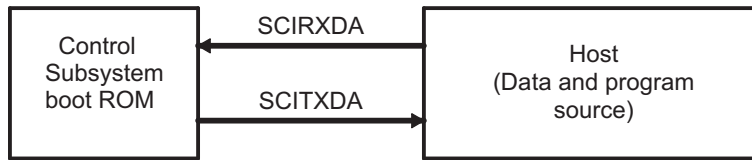
Figure 3-9. Overview of CopyData Function



3.25.4 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 same data flow as outlined in [Example 3-2](#).

Figure 3-10. Overview of SCI Bootloader Operation



The device communicates with the external host device 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 28x application will echo back the 8-bit character received to the host. In this manner, the host can perform checks that each character was received by the 28x application.

At higher baud rates, the slew rate of the incoming data bits can be effected 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 28x SCI bootloader using a lower baud rate.
2. Load the incoming 28x application or custom loader at this lower baud rate.
3. The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired high baud rate.

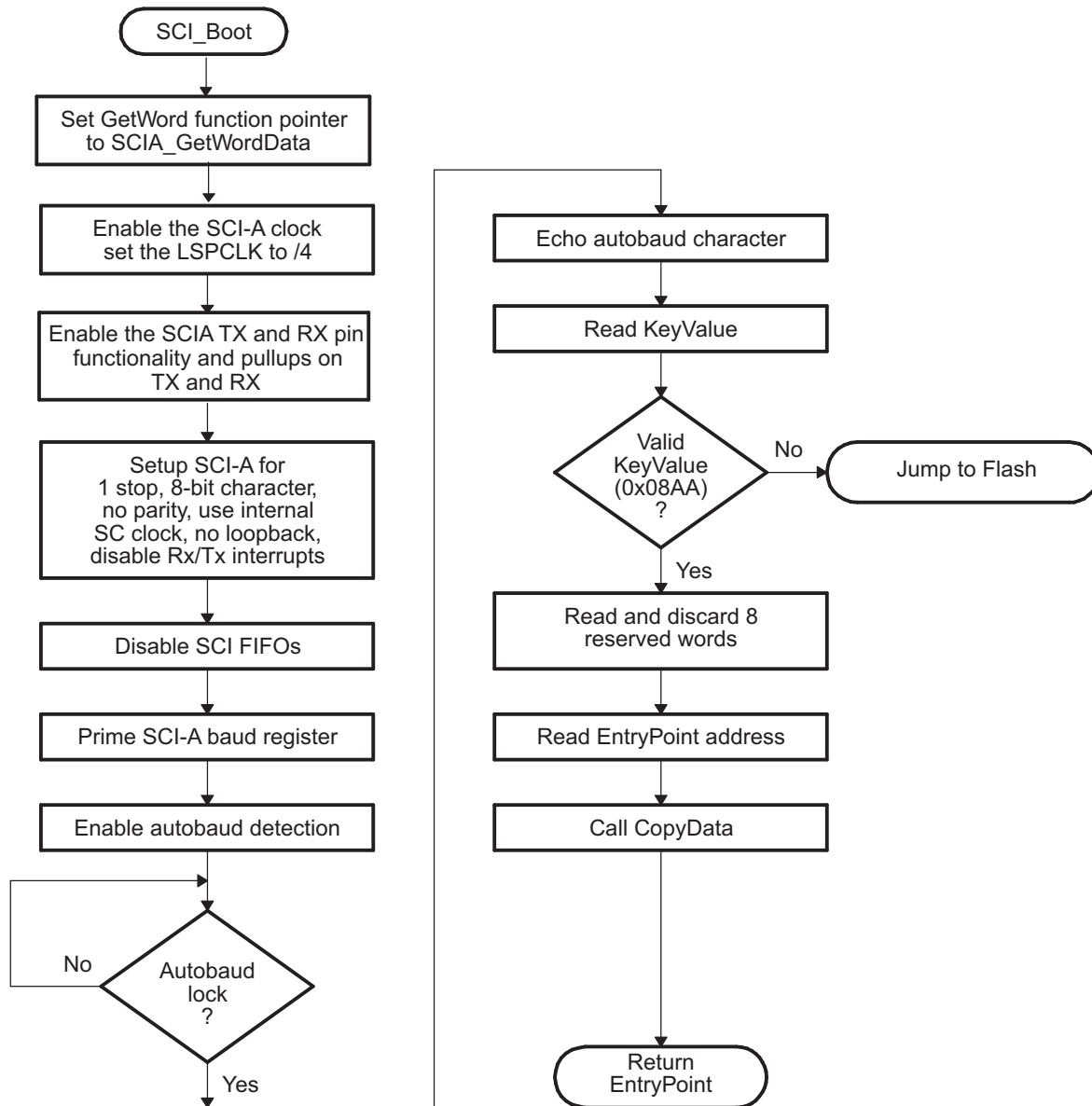
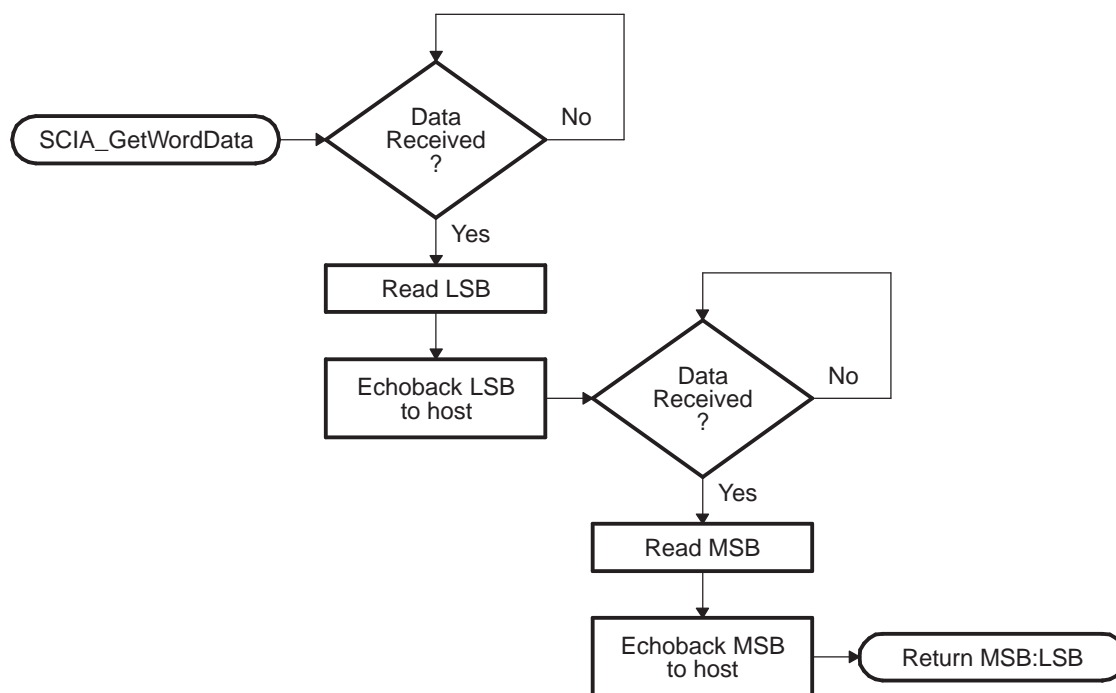
Figure 3-11. Overview of SCI Boot Function


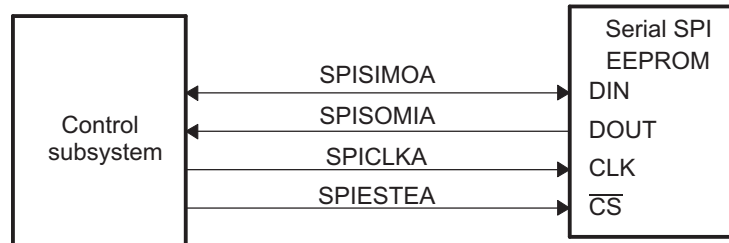
Figure 3-12. Overview of SCI_GetWordData Function



3.25.5 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 3-13. The SPI bootloader supports an 8-bit data stream. It does not support a 16-bit data stream.

Figure 3-13. 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 3-20. 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 GPIO19 (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
- 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 3-14. Data Transfer From EEPROM Flow

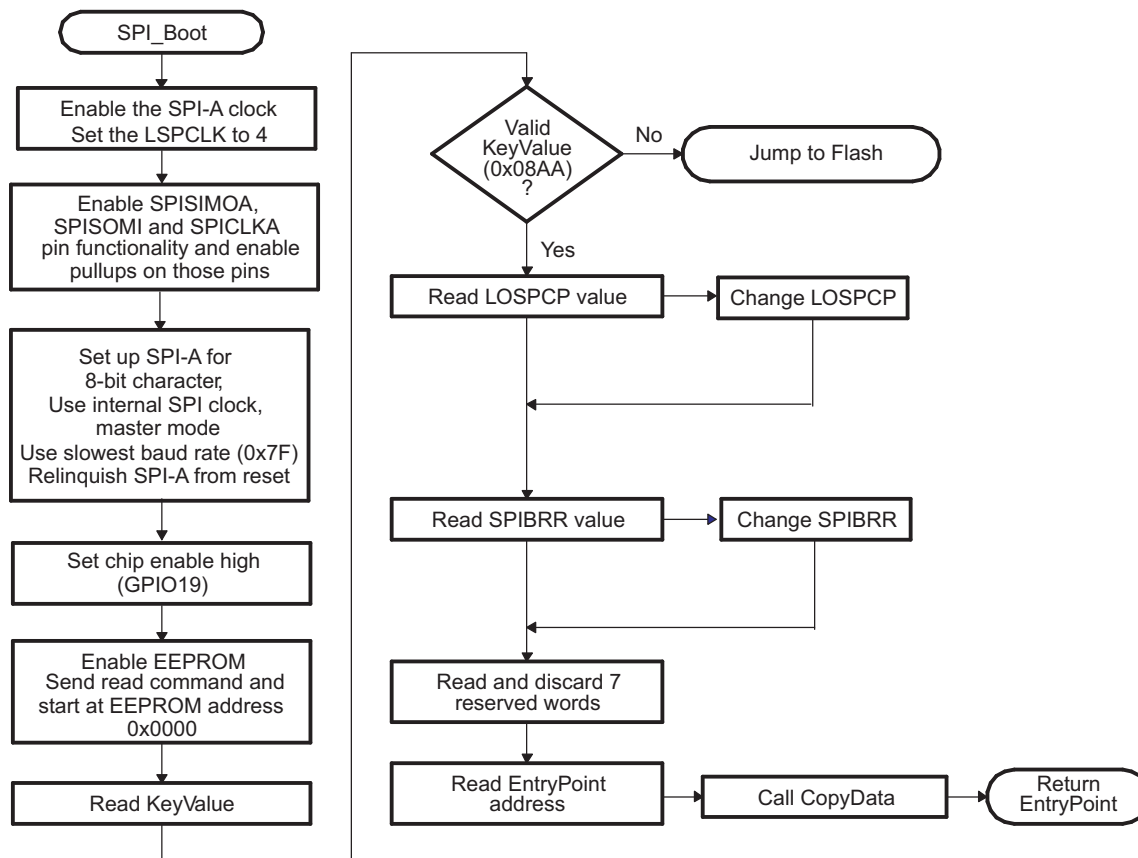
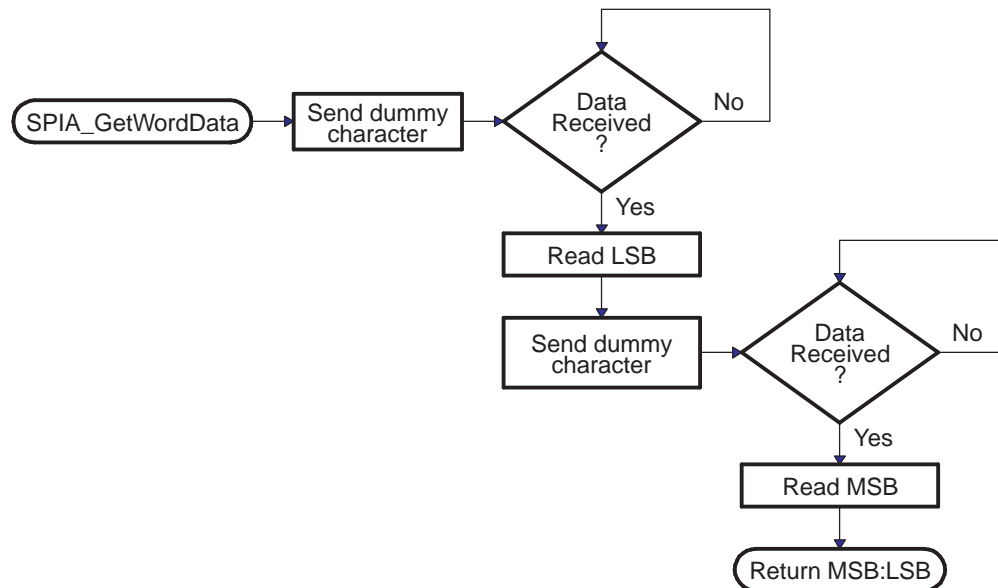
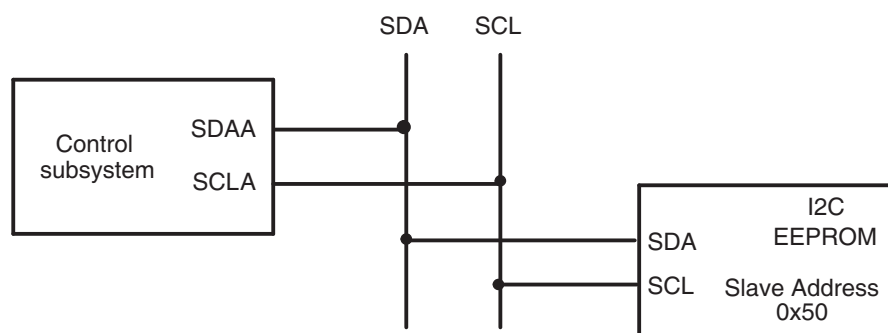


Figure 3-15. Overview of SPIA_GetWordData Function


3.25.6 I2C Boot Mode

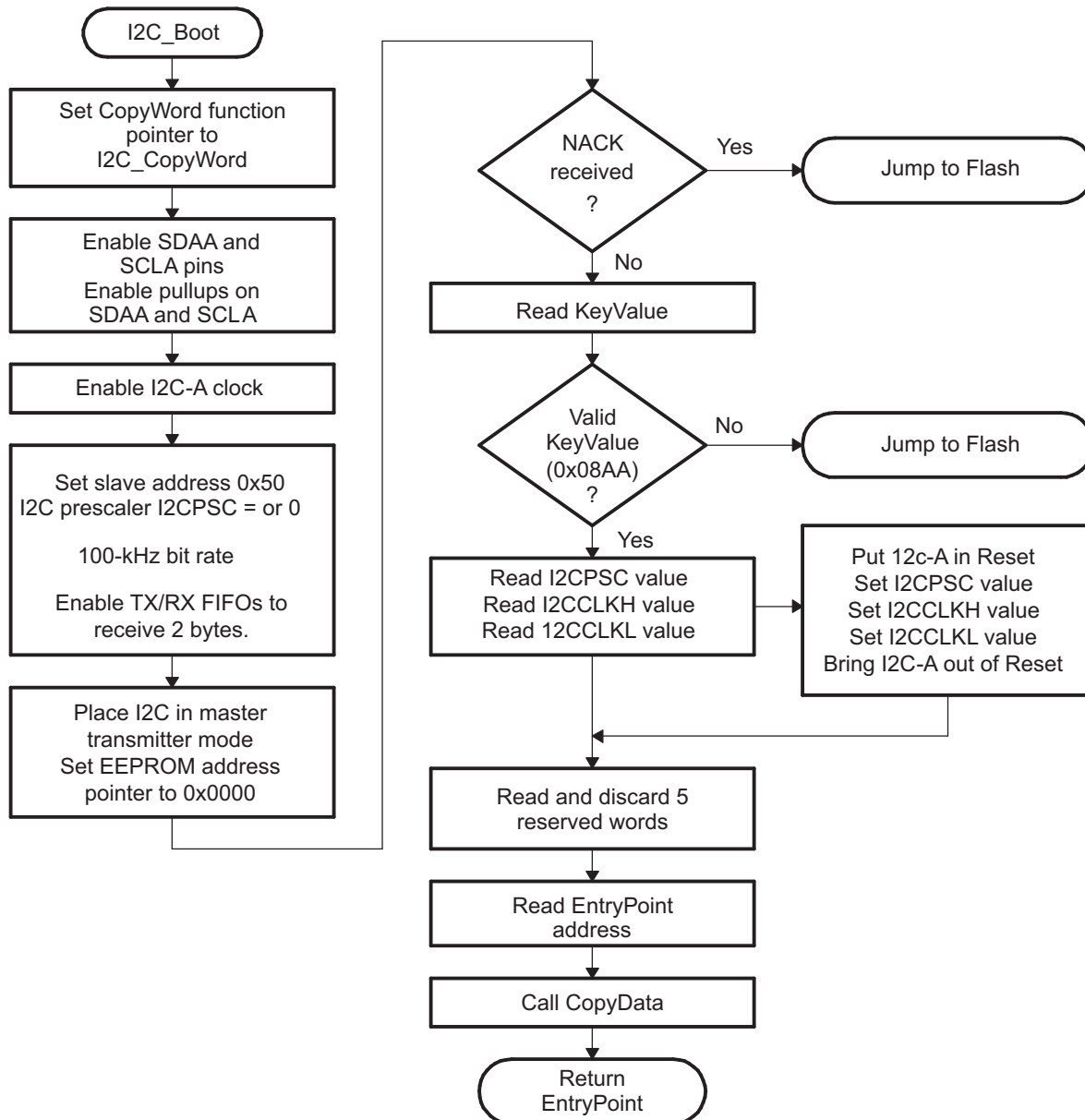
The I2C bootloader expects an 8-bit wide I2C-compatible EEPROM device to be present at address 0x50 on the I2C-A bus as indicated in [Figure 3-16](#). The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.

Figure 3-16. 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 3-17. 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.

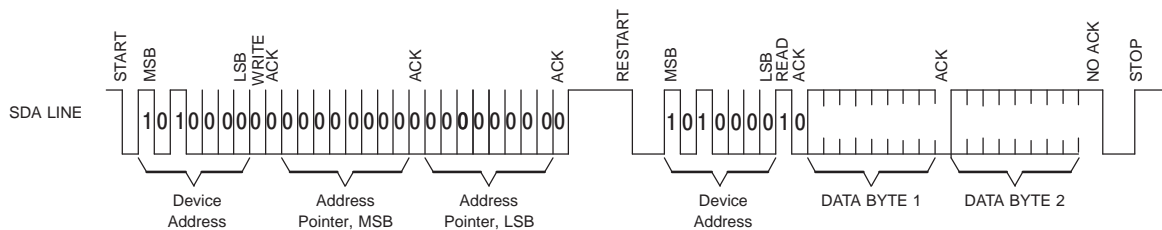
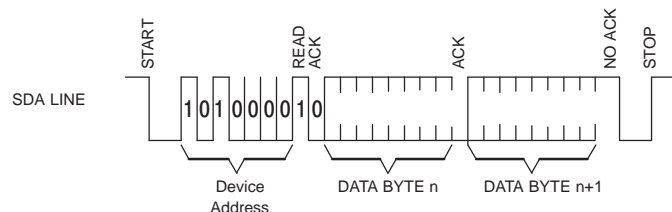
Arbitration, bus busy, and slave signals are not checked. Therefore, no other master is allowed to control the bus during this initialization phase. If the application requires another master during I2C boot mode, that master must be configured to hold off sending any I2C messages until the application 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, code will 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 13-1](#) shows the 8-bit data stream used by the I2C.

Table 3-21. 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

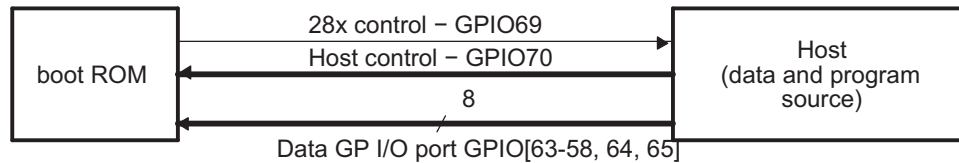
The I2C EEPROM protocol required by the I2C bootloader is shown in [Figure 3-18](#) and [Figure 3-19](#). The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA) from it, is shown in [Figure 3-18](#). All subsequent reads are shown in [Figure 3-19](#) and are read two bytes at a time.

Figure 3-18. Random Read

Figure 3-19. Sequential Read


3.25.7 Parallel Boot Mode

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO0 -GPIO5, GPIO8-GPIO9 to internal memory. Each value is 8 bits long and follows the same data flow as outlined in [Figure 3-20](#).

Figure 3-20. Overview of Parallel GPIO Bootloader Operation



The control subsystem communicates with the external host device by polling/driving the GPIO70 and GPIO69 lines. The handshake protocol shown in [Figure 3-21](#) must be used to successfully transfer each word via GPIO [63-58,64,65]. 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[63-58,64,65].

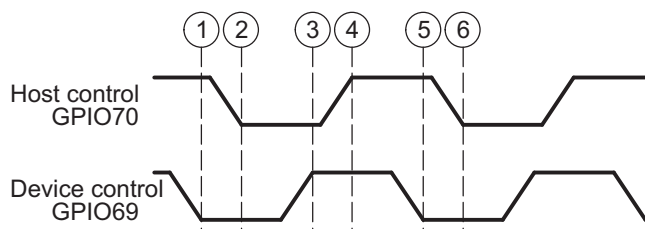
The 8-bit data stream is shown in [Table 3-22](#).

Table 3-22. Parallel GPIO Boot 8-Bit Data Stream

Bytes	GPIO[9,8,5:0] (Byte 1 of 2)	GPIO[9,8,5:0] (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 = 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
.	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 GPIO69 pin low. The host load then initiates the data transfer by pulling the GPIO70 pin low. The complete protocol is shown in Figure 3-21:

Figure 3-21. Parallel GPIO Bootloader Handshake Protocol



1. The device indicates it is ready to start receiving data by pulling the GPIO69 pin low.
2. The bootloader waits until the host puts data on GPIO [63-58,64,65]. The host signals to the device that data is ready by pulling the GPIO70 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO69 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO70 high.
5. The device again indicates it is ready for more data by pulling the GPIO69 pin low.

This process is repeated for each data value to be sent.

Figure 3-22 shows an overview of the Parallel GPIO bootloader flow.

Figure 3-22. Parallel GPIO Mode Overview

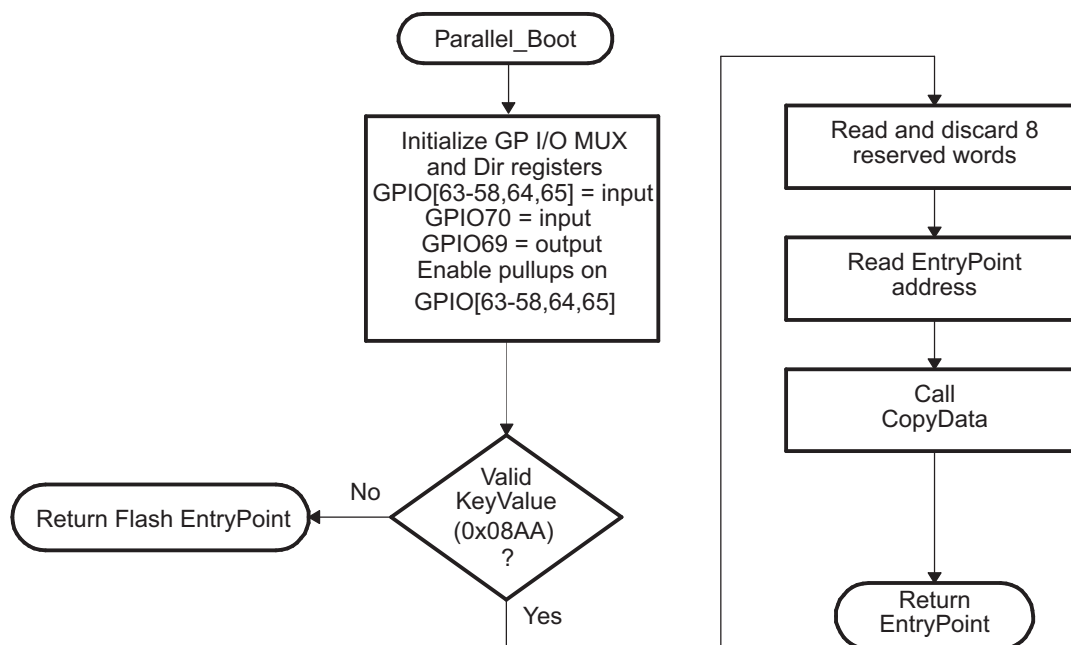


Figure 3-23 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host will wait for the device and the device will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower than the device.

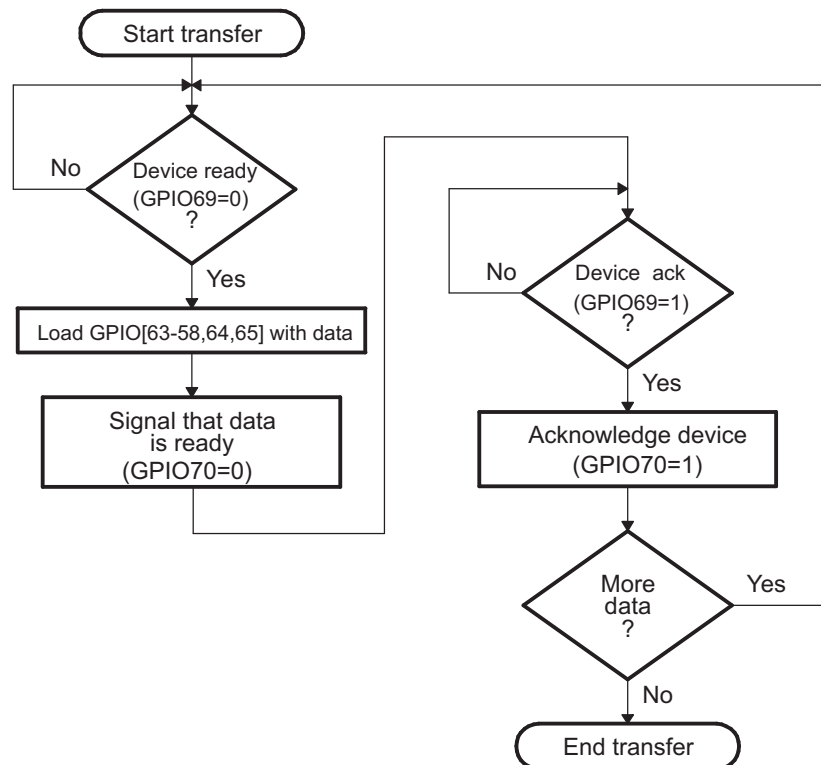
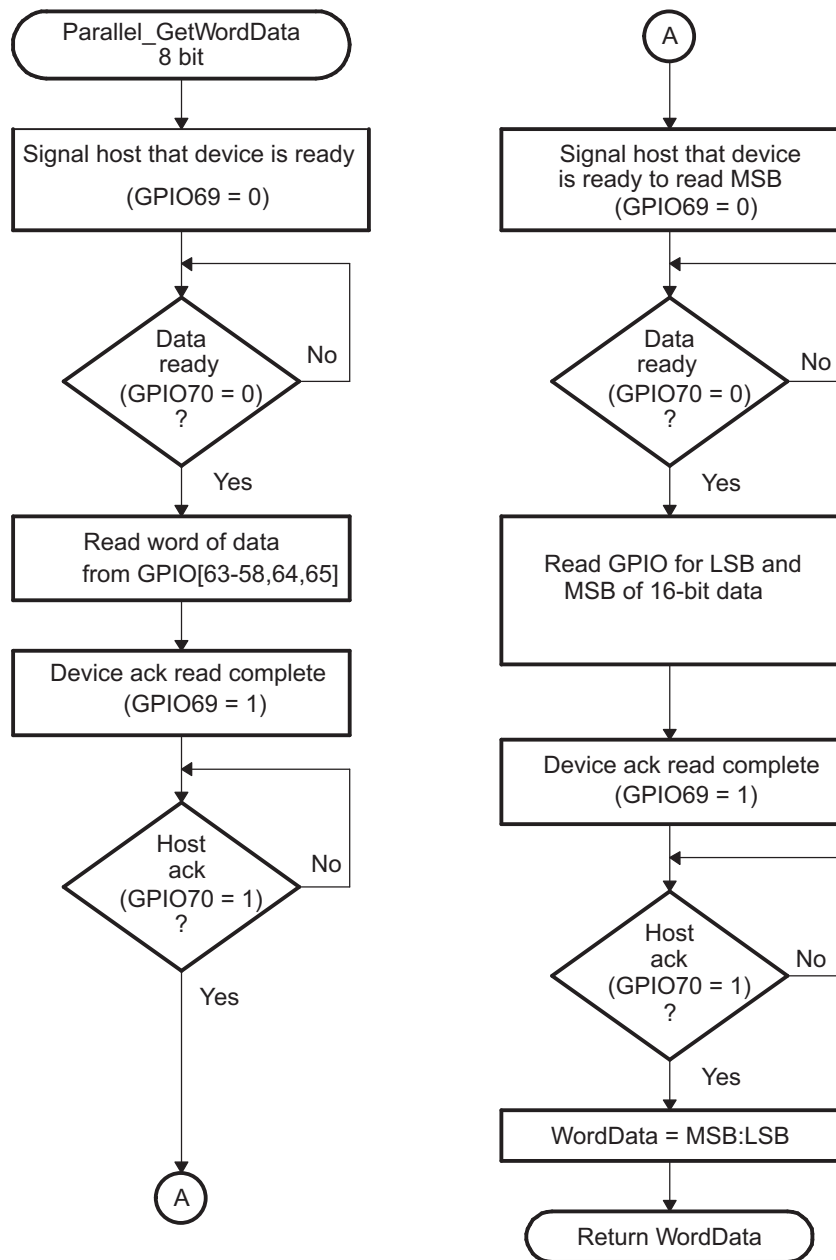
Figure 3-23. Parallel GPIO Mode - Host Transfer Flow


Figure 3-24 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 3-24, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with GPIO9 in bit position 7 and GPIO8 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). 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.

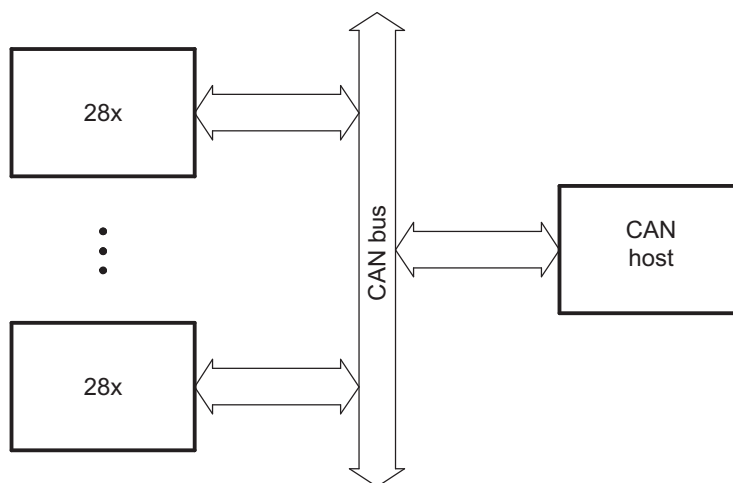
Figure 3-24. 8-Bit Parallel GetWord Function



3.25.8 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 3-25. 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 3-23](#).

Table 3-23. Bit-Rate Value for Internal Oscillators

OSCCLK	SYSCCLKOUT	Bit Rate
20 MHz	10 MHz	100 kbps

The SYSCCLKOUT 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.

NOTE: The CPU1 CAN boot loader uses XTAL as the bit clock source and INTOSC2 as the system clock source. The CPU2 CAN boot loader does not change either clock source, so CPU1 must configure the clock sources before starting the CPU2 CAN boot loader.

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 3-24](#):

Table 3-24. 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

Table 3-24. CAN 8-Bit Data Stream (continued)

Bytes		Byte 1 of 2	Byte 2 of 2	Description
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
.		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

3.25.9 USB Boot Mode

In USB boot mode, the device will enumerate with vendor ID 0x1cbe and product ID 0x00ff. The device descriptor and interface descriptor both show the class as 0xFF (vendor-specific), the subclass as 0x00, and the protocol as 0x00. After enumeration, the device will wait for data. Data should be sent via bulk OUT transfers to endpoint 1. The data is interpreted as a series of 8-bit bytes in the standard data stream format described in [Section 3.25.1](#), shown here in [Table 3-25](#). No reserved bytes are used. Once the data transfer is complete (block size of 0x0000 sent), the device will disconnect from the USB bus, allowing other software to make use of the module if desired. [Figure 3-26](#) illustrates the flow for USB boot mode.

Figure 3-26. USB Boot Flow

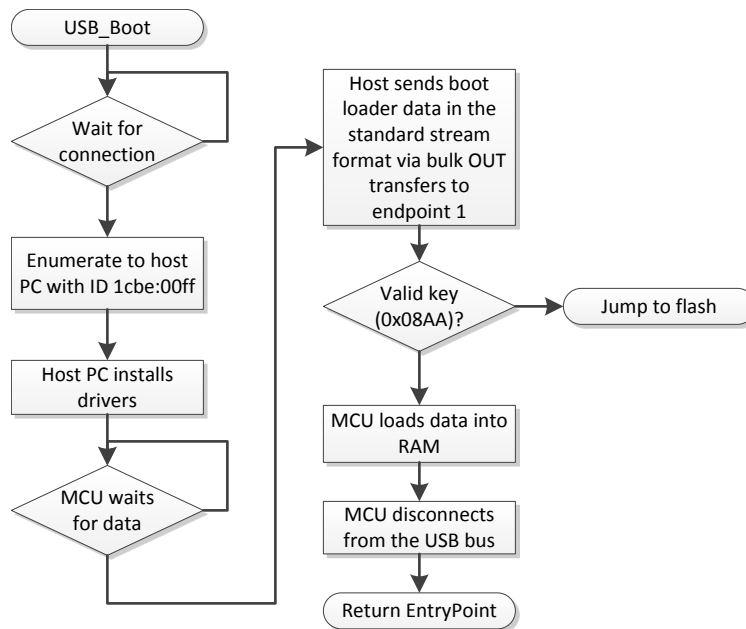


Table 3-25. USB 8-Bit Data Stream

Bytes	First Byte (LSB)	Second Byte (MSB)	Description
1 2	AA	08	0x08AA (KeyValue for memory width = 16bits)
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 3-25. USB 8-Bit Data Stream (continued)

Bytes	First Byte (LSB)	Second Byte (MSB)	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

Implementing PC-side USB software is not trivial. It is recommended to use the TI-provided tools and drivers to load data in USB boot mode. Hex and binary files for loader tools can be generated from COFF (.out) files using the hex2000 tool. To produce a plain binary file in the boot loader format, use the following command line:

```
hex2000 -boot -b Program_to_Load.out -o Binary_Loader_Data.dat
```

For more information on hex2000, please see the *TMS320C28x Assembly Language Tools User's Guide* ([SPRU513](#)).

3.26 CLA Data ROM

Both CPU1 and CPU2 on the device support CLA data ROM and this data ROM has some Math tables for CLA application usage. This section provides details about this.

3.26.1 CPU1/CPU2 CLA Data ROM

3.26.1.1 CLA Data ROM Map

CLAlmathTables			
*	1	01001870	00000784
		01001870	0000018c CLAAatanTable.obj (CLAlmathTables)
		010019fc	0000018a CLAAasineTable.obj (CLAlmathTables)
		01001b86	00000184 CLAAcosineTable.obj (CLAlmathTables)
		01001d0a	00000156 CLASinCosTable.obj (CLAlmathTables)
		01001e60	000000d2 CLAlnTable.obj (CLAlmathTables)
		01001f32	000000c2 CLAexpTable.obj (CLAlmathTables)
\$fill1006	1	01001ff4	00000006
		01001ff4	00000006 --HOLE-- [fill = ffff]
.CROMVersion			
*	1	01001ffa	00000002
		01001ffa	00000002 cla_rom_version.obj (.CROMVersion)
\$fill1007	1	01001ffc	00000004
		01001ffc	00000004 --HOLE-- [fill = ffff]

3.26.1.2 CLA Data ROM Tables

Table 3-26. CLA Data ROM Tables

	Start Address From CLA in Hex	Start Address From CPU in Hex
_CLAatan2HalfPITable	F870	01001870
_CLAINV2PI	F874	01001874
_CLAatan2Table	F876	01001876
_CLAasinHalfPITable	F9fc	010019fc

Table 3-26. CLA Data ROM Tables (continued)

	Start Address From CLA in Hex	Start Address From CPU in Hex
_CLAatan2TableEnd	F9fc	010019fc
_CLAasinTable	Fa00	01001a00
_CLAacosinHalfPITable	Fb86	01001b86
_CLAasinTableEnd	Fb86	01001b86
_CLAacosinTable	Fb8a	01001b8a
_CLAacosinTableEnd	Fd0a	01001d0a
_CLAsinTable	Fd0a	01001d0a
_CLAsincosTable	Fd0a	01001d0a
_CLAsincosTable_Sin0	Fd0a	01001d0a
_CLAcosTable	Fd4a	01001d4a
_CLAsincosTable_Cos0	Fd4a	01001d4a
_CLAsinTableEnd	Fe0a	01001e0a
_CLAcosTableEnd	Fe4c	01001e4c
_CLAsincosTable_TABLE_SIZE	Fe4c	01001e4c
_CLAsincosTable_TABLE_SIZEDivTwoPi	Fe4e	01001e4e
_CLAsincosTable_TwoPiDivTABLE_SIZE	Fe50	01001e50
_CLAsincosTable_TABLE_MASK	Fe52	01001e52
_CLAsincosTable_Coef0	Fe54	01001e54
_CLAsincosTable_Coef1	Fe56	01001e56
_CLAsincosTable_Coef1_pos	Fe58	01001e58
_CLAsincosTable_Coef2	Fe5a	01001e5a
_CLAsincosTable_Coef3	Fe5c	01001e5c
_CLAsincosTable_Coef3_neg	Fe5e	01001e5e
_CLALNV2	Fe60	01001e60
_CLAsincosTableEnd	Fe60	01001e60
_CLALNVe	Fe62	01001e62
_CLALNV10	Fe64	01001e64
_CLABIAS	Fe66	01001e66
_CLALN_TABLE_MASK1	Fe68	01001e68
_CLALN_TABLE_MASK2	Fe6a	01001e6a
_CLALnTable	Fe6c	01001e6c
_CLAINV1	Ff32	01001f32
_CLALnTableEnd	Ff32	01001f32
_CLAINV2	Ff34	01001f34
_CLAINV3	Ff36	01001f36
_CLAINV4	Ff38	01001f38
_CLAINV5	Ff3a	01001f3a
_CLAINV6	Ff3c	01001f3c
_CLAINV7	Ff3e	01001f3e
_CLALOG10	Ff40	01001f40
_CLAExpTable	Ff42	01001f42
_CLAExpTableEnd	Fff4	01001ff4
CROM VERSION	FFFA	01001FFA (2 16-bit words) .word 0x0100 ; Boot ROM Version v1.0 .word 0x0413 ; Month/Year: (ex: 0x0109 = 1/09 = Jan 2009)

3.27 Secure ROM Contents

Secure ROM on CPU1/CPU2 has no libraries programmed on REV0 of the devices, but below functions are provided in Zx-SCC sections of secure ROM.

```

Z1_SCC_ROM      : origin = 0x3F0000, length = 0x000800, fill = 0xFFFF      /* Zone 1 Safe-
Copy Code Secure ROM */
Z2_SCC_ROM      : origin = 0x3F0800, length = 0x000800, fill = 0xFFFF      /* Zone 2 Safe-
Copy Code Secure ROM */
Z1_SECURE_ROM   : origin = 0x3F1008, length = 0x006FF8, fill = 0xFFFF      /* Z1 Secure ROM
*/

```

3.27.1 CPU1.Secure ROM

Information to be provided.

3.27.1.1 SafeCopyCode

Information to be provided.

3.27.1.2 SafeCRCCalc

Information to be provided.

3.27.2 CPU2.Secure ROM

Information to be provided.

3.27.2.1 SafeCopyCode

Information to be provided.

3.27.2.2 SafeCRCCalc

Information to be provided.

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
4.1 Introduction	573
4.2 Architecture	574
4.3 Pipeline Timing and Throughput	582
4.4 CPU Arbitration	583
4.5 Channel Priority	584
4.6 Address Pointer and Transfer Control.....	585
4.7 Overrun Detection Feature.....	589
4.8 Register Descriptions.....	591

4.1 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 interrupt trigger source, there is no mechanism within the module itself to start memory transfers periodically. The interrupt trigger source for each of the six DMA channels 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.

DMA features include:

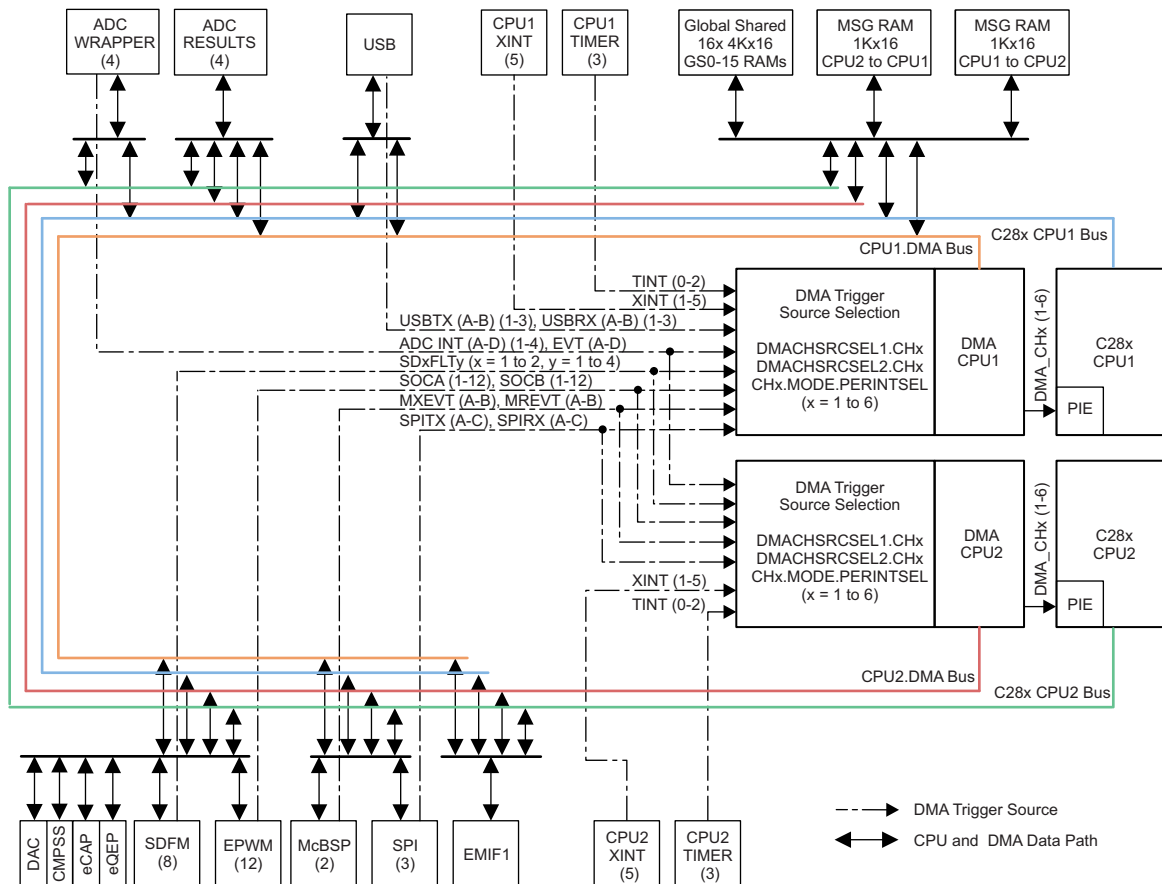
- Six channels with independent PIE interrupts
- Peripheral interrupt trigger sources
 - ADC interrupts and EVT signals
 - McBSPx transmit and receive
 - External Interrupts
 - CPU Timers
 - EPWMxSOC signals
 - SPIx transmit and receive
 - USBx transmit and receive
 - Sigma Delta filter module
 - Software trigger
- Data sources and destinations:
 - GSx RAM
 - CPU message RAM (IPC RAM)
 - USB RAM
 - ADC memory-bus mapped result registers
 - ePWMx
 - SPI, McBSP, EMIF
- Word Size: 16-bit or 32-bit (SPI and McBSP limited to 16-bit)
- Throughput: 4 cycles/word without arbitration

4.2 Architecture

4.2.1 Block Diagram

Figure 4-1 shows a device-level block diagram of the DMA.

Figure 4-1. DMA Block Diagram



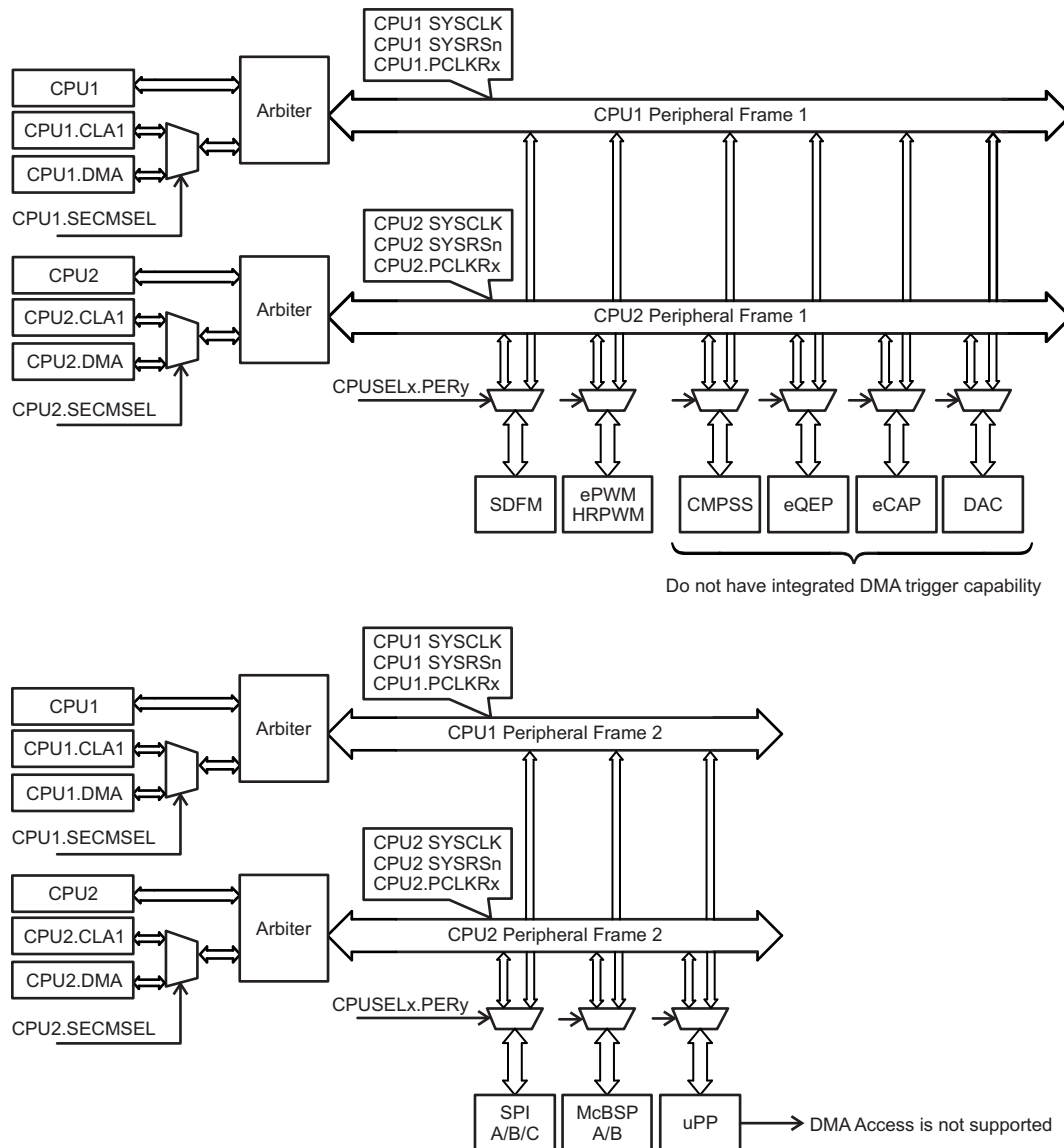
4.2.2 Common Peripheral Architecture

There are two CPU subsystems; the CPU1 subsystem and CPU2 subsystem, with each containing a CLA and a DMA. The architecture allows several peripherals to be common between the two subsystems. Based on application need, these common peripherals can be attached to one of the two subsystems. Figure 4-2 shows how the CPUs and subsystems can be connected to the peripherals on peripheral frames 1 and 2. The clock, clock-enable, and reset muxing for the common peripherals are described in detail in other sections of this document.

Refer to Section 4.4 for more details on the arbitration scheme for all masters.

NOTE: If the CPU and DMA make an access to the same peripheral frame in the same cycle, the DMA has priority and the CPU is stalled.

Figure 4-2. Common Peripheral Architecture



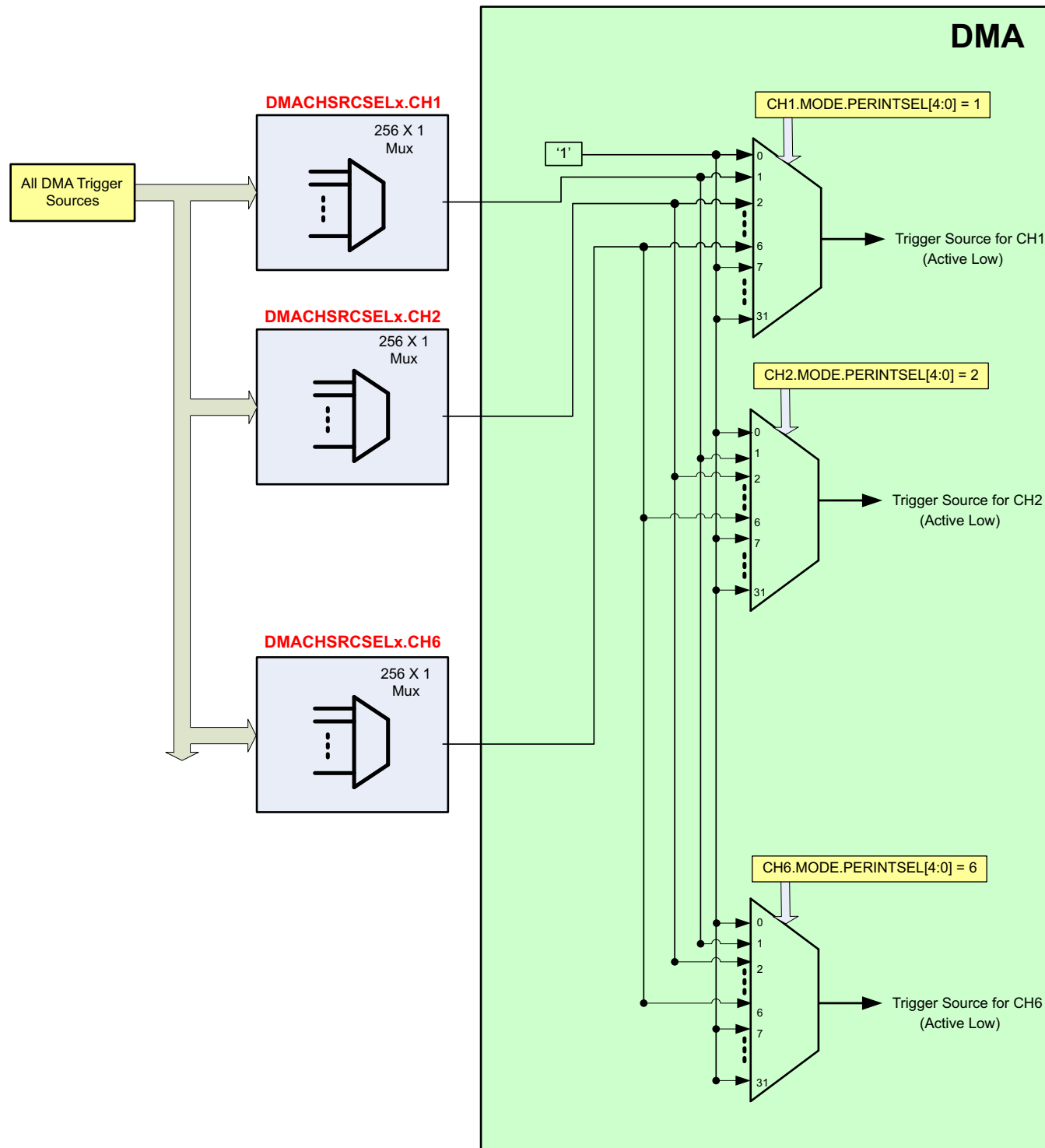
A CPUSEL bit associated with each peripheral defines whether the peripheral belongs to the CPU1 or CPU2 subsystem. If a peripheral belongs to a CPU subsystem, it can be accessed by the CPU and one of the secondary masters (DMA or CLA1). Refer to CPUSELx register definition for more details. The secondary master is statically selected using the SECMSEL register mapped to the respective CPU. Refer to CPUx.SECMSEL register definition for more details. If a secondary master is not selected, all writes from that master are ignored and all reads return 0x0 to any of the peripherals.

Similarly, if a peripheral does not belong to a CPU subsystem (as defined by the associated CPUSEL bit), all writes to that peripheral are ignored and all reads to that peripheral return 0x0 from any of the masters belonging to the unselected CPU subsystem. Note that since the arbiter has no knowledge regarding the ownership of individual peripherals (as can be seen from Figure 4-2), arbitration will still happen even if C28x or the selected secondary master tries to access a peripheral which does not belong to its CPU subsystem. See [CPU Arbitration](#) for more information.

4.2.3 Peripheral Interrupt Event Trigger Sources

As shown in [Figure 4-3](#) the peripheral interrupt event trigger can be independently configured as any of the sources from the DMACHSRCSELx register for each of the six DMA channels. Included in these 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. The DMA trigger source is selected in the DMACHSRCSELx register ([Table 2-16](#)), for each channel. The PERINTSEL in the MODE register of each channel should be set to the channel number (for example, CH2.MODE.PERINTSEL[4:0] = 2). An active peripheral interrupt trigger will be latched into the PERINTFLG bit of the CONTROL register, and if the respective interrupt and DMA channel is enabled (see the MODE.CHx[PERINTE] and CONTROL.CHx[RUNSTS] bits), it will be serviced by the DMA channel. 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 4-3. DMA Trigger Architecture



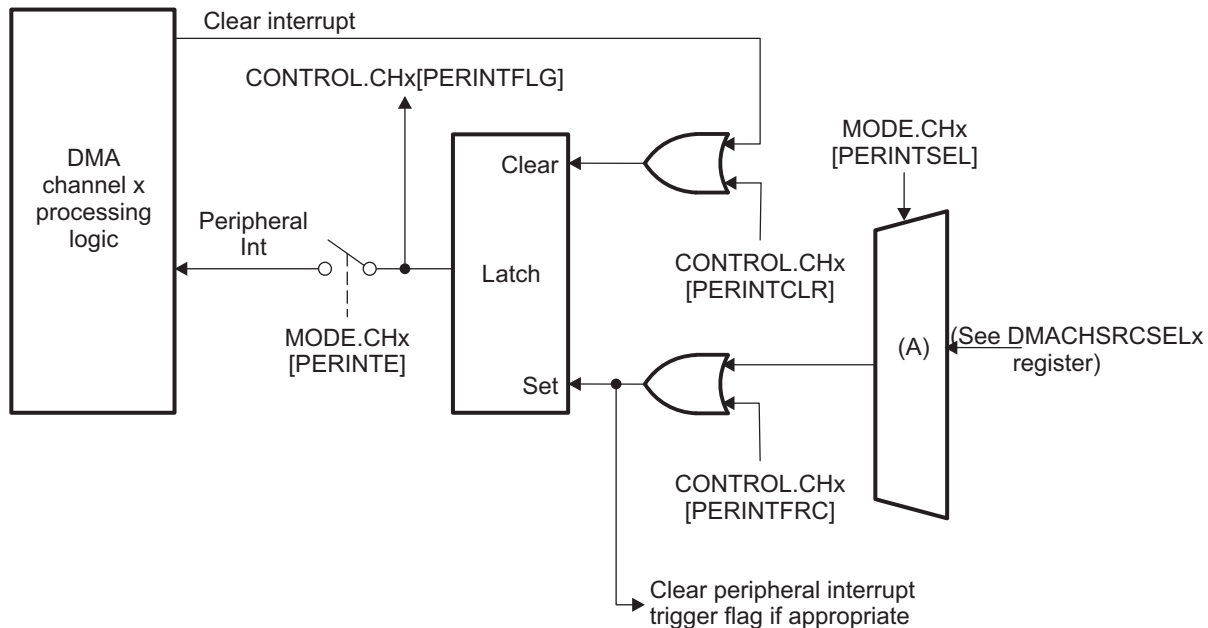
NOTE: To use the system level DMA Trigger source selection, the DMA internal trigger source selection configuration for each channel should be done using the DMACHSRCSELx register (Table 2-16), and the CHx.MODE.PERINTSEL register as shown here. See Table 4-1 or the DMACHSRCSELx register definition for 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 interrupt trigger sets a channel's `PERINTFLG` bit, the bit stays 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 interrupt trigger is generated while a burst is in progress, the burst will complete before responding to the new interrupt trigger (after proper prioritization). If a third interrupt trigger occurs before the pending interrupt is serviced, an error flag is set in the `CONTROL.CHx[OVRFLG]` bit. If a peripheral interrupt trigger occurs at the same time as the latched flag is being cleared, the peripheral interrupt trigger has priority and the `PERINTFLG` will remain set.

Figure 4-4 shows a diagram of the trigger select circuit. See the `DMACHSRCSELx` register, (Table 2-16) description for the complete list of peripheral interrupt trigger sources.

Figure 4-4. Peripheral Interrupt Trigger Input Diagram



A See Figure 4-3.

Table 4-1 shows the interrupt trigger source options that are available for each channel.

Table 4-1. Peripheral Interrupt 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	ADCB EVT
11	ADCC.1
12	ADCC.2
13	ADCC.3
14	ADCC.4
15	ADCCEVT
16	ADCD.1
17	ADCD.2
18	ADCD.3
19	ADCD.4
20	ADCDEV T
21	No Peripheral
22	No Peripheral
23	No Peripheral
24	No Peripheral
25	No Peripheral
26	No Peripheral
27	No Peripheral
28	No Peripheral
29	XINT1
30	XINT2
31	XINT3
32	XINT4
33	XINT5
34	No Peripheral
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

Table 4-1. Peripheral Interrupt Trigger Source Options (continued)

Select Value (8-bit)	DMA ChTrigger Source
48	EPWM7.SOCA
49	EPWM7.SOCB
50	EPWM8.SOCA
51	EPWM8.SOCB
52	EPWM9.SOCA
53	EPWM9.SOCB
54	EPWM10.SOCA
55	EPWM10.SOCB
56	EPWM11.SOCA
57	EPWM11.SOCB
58	EPWM12.SOCA
59	EPWM12.SOCB
60	No Peripheral
61	No Peripheral
62	No Peripheral
63	No Peripheral
64	No Peripheral
65	No Peripheral
66	No Peripheral
67	No Peripheral
68	TINT0
69	TINT1
70	TINT2
71	MXEVTA
72	MREVTA
73	MXEVTB
74	MREVTB
75	No Peripheral
76	No Peripheral
77	No Peripheral
78	No Peripheral
79	No Peripheral
80	No Peripheral
81	No Peripheral
82	No Peripheral
83	No Peripheral
84	No Peripheral
85	No Peripheral
86	No Peripheral
87	No Peripheral
88	No Peripheral
89	No Peripheral
90	No Peripheral
91	No Peripheral
92	No Peripheral
93	No Peripheral
94	No Peripheral

Table 4-1. Peripheral Interrupt Trigger Source Options (continued)

Select Value (8-bit)	DMA ChTrigger Source
95	SD1FLT1
96	SD1FLT2
97	SD1FLT3
98	SD1FLT4
99	SD2FLT1
100	SD2FLT2
101	SD2FLT3
102	SD2FLT4
103	No Peripheral
104	No Peripheral
105	No Peripheral
106	No Peripheral
107	No Peripheral
108	No Peripheral
109	SPITXDMAA
110	SPIRXDMAA
111	SPITXDMAB
112	SPIRXDMAB
113	SPITXDMAC
114	SPIRXDMAC
115	No Peripheral
116	No Peripheral
117	No Peripheral
118	No Peripheral
119	No Peripheral
120	No Peripheral
121	No Peripheral
122	No Peripheral
123	No Peripheral
124	No Peripheral
125	No Peripheral
126	No Peripheral
127	No Peripheral
128	No Peripheral
129	No Peripheral
130	No Peripheral
131	USBA_EPx_RX1
132	USBA_EPx_TX1
133	USBA_EPx_RX2
134	USBA_EPx_TX2
135	USBA_EPx_RX3
136	USBA_EPx_TX3
137:255	No Peripheral

4.2.4 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 4.4](#).

4.3 Pipeline Timing and Throughput

The DMA module consists of a 4-stage pipeline as shown in [Figure 4-5](#). The one exception to this is when a DMA channel is configured to have the McBSP as its data source. A read of a McBSP DRR register stalls the DMA bus for one cycle during the read portion of the transfer, as shown in [Figure 4-6](#).

Figure 4-5. 4-Stage Pipeline DMA Transfer

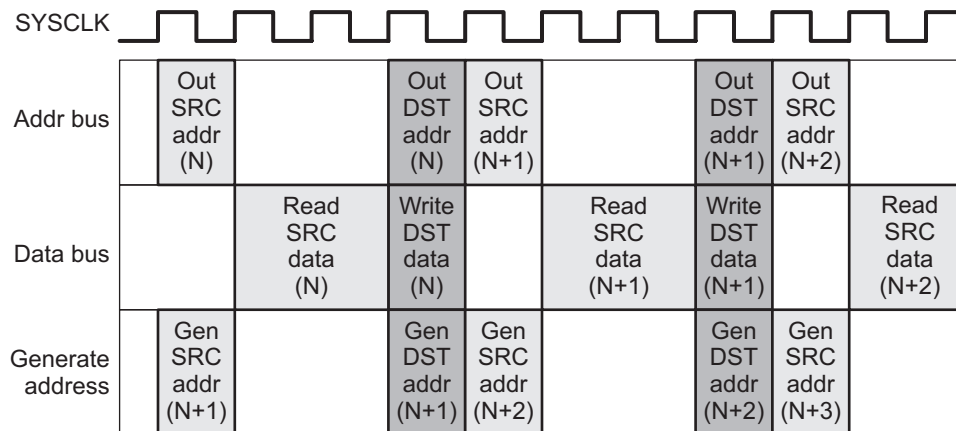
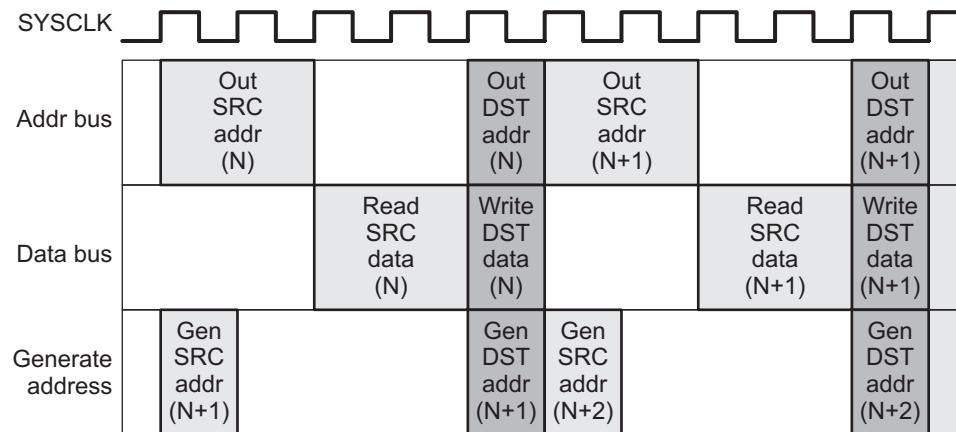


Figure 4-6. 4-Stage Pipeline With One Read Stall (McBSP as source)



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 4.4](#))
- 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 GS15 RAM, a channel can be configured to transfer 8 bursts of 16 words/burst. This will give:

$$8 \text{ bursts} * [(4 \text{ cycles/word} * 16 \text{ words/burst}) + 1] = 520 \text{ cycles}$$

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:

$$8 \text{ bursts} * [(4 \text{ cycles/word} * 8 \text{ words/burst}) + 1] = 264 \text{ cycles}$$

4.4 CPU Arbitration

Typically, DMA activity is independent of the CPU activity. When the DMA and the CPU access a peripheral register within the same peripheral frame concurrently, an arbitration procedure will occur. Any combined accesses between the different interfaces, or where the CPU access is outside of the interface that the DMA is accessing, do not create a conflict.

Conflict Example: the CPU is accessing SPI FIFO while the DMA is simultaneously accessing McBSP, it will create a conflict because both SPI and McBSP reside in a common interface (peripheral frame 2).

Non-conflict Example: the CPU is accessing shared ePWM while the DMA is accessing message McBSP, there will be no conflict, since these two peripherals are located in different interfaces (peripheral frame 1 and peripheral frame 2).

The interfaces which internally contain conflicts are:

- Global shared RAMs (GSRAMs)
- CPU message RAM (IPC MSG RAMs)
- USB registers
- EMIF1 registers
- Peripheral frame 1 (ePWM, HRPWM, eCAP, eQEP, DAC, CMPSS, and SDFM)
- Peripheral frame 2 (McBSP, SPI, and uPP)

The ADC result registers are duplicated for each bus master. Therefore, the CPU, DMA, and CLA can all simultaneously read these registers with no stalls for any master.

If the CPU and the DMA make an access to the same peripheral frame in the same cycle, the DMA has priority and the CPU is stalled.

A DMA transfer consists of four phases: send source address, read source data, send destination address, and write destination data (see [Section 4.3](#)). In the case of a block DMA transfer to and from the same memory interface the CPU is trying to access, the arbitration will stall CPU access until the DMA completes an access, not the entire transfer. This arbitration is based on a round robin priority scheme described in [Section 4.5](#).

The following priority schemes are implemented for the various interfaces on the device.

- On the peripheral register interface, the fixed priority scheme is:
 - CLA/DMA Write
 - CLA/DMA Read
 - CPU Write
 - CPU Read
- For ADC result registers (refer to the *ADC* chapter for more detailed information):
 - Parallel access is possible from all masters
- For GSxRAM/IPC MSG RAM (refer to the *System Control* chapter for more detailed information):
 - Round-robin

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. For this reason, it is advised not to mix such CPU accesses with DMA accesses to the same locations.

4.5 Channel Priority

Two priority schemes exist when determining channel priority: Round-robin mode and Channel 1 high-priority mode.

4.5.1 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.

4.5.2 Channel 1 High Priority Mode

In this mode, if a CH1 trigger occurs, the current word transfer or the current + 1 word transfer (depends on which phase of the current DMA transfer the new CH1 trigger occurred) on any other channel is completed (not the complete burst), execution is halted, and CH1 is serviced for the complete burst count. When the CH1 burst is complete, execution returns to the channel that was active when the CH1 trigger occurred. All other channels have equal priority and each enabled channel is serviced in round-robin fashion as follows:

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.

4.6 Address Pointer and Transfer Control

The DMA state machine is, at its most basic level, two nested loops. The inner loop transfers a burst of data when a peripheral interrupt trigger is received. A burst is the smallest amount of data that can be transferred at one time and its size is defined by the BURST_SIZE register for each channel. The BURST_SIZE register allows a maximum of 32 sixteen-bit words to be transferred in one burst. The outer loop, whose size is set by the TRANSFER_SIZE register for each channel, defines how many bursts are performed in the entire transfer. Since TRANSFER_SIZE is a 16-bit register, the total size of a transfer allowed is well beyond any practical requirement. One CPU interrupt is generated, if enabled, for each transfer. This interrupt can be configured to occur at the beginning or the end of the transfer via the MODE.CHx[CHINTMODE] bit.

In the default setting of the MODE.CHx[ONESHOT] bit, the DMA transfers one burst of data each time a peripheral interrupt 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. If a transfer of more than the maximum number of words per burst is desired for a single trigger, the MODE.CHx[ONESHOT] bit can be set to complete the entire transfer when triggered. Care is advised when using this mode, since this can create a condition where one trigger uses up the majority of the DMA bandwidth.

Each DMA channel contains a shadowed address pointer for the source and the destination address. These pointers, SRC_ADDR and DST_ADDR, can be independently controlled during the state machine operation. At the beginning of each transfer, the shadowed version of each pointer is copied into its respective 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 SRC/DST_ADDR register. During the transfer loop, after each burst is complete, there are two methods that can be used to modify the active address pointer. The first, and default, method is by adding the signed value contained in the SRC/DST_TRANSFER_STEP register to the appropriate pointer. The second is via a process called wrapping, where a wrap address is loaded into the active address pointer. When a wrap procedure occurs, the associated SRC/DST_TRANSFER_STEP register has no effect.

Address wrapping occurs when a number of bursts specified by the appropriate SRC/DST_WRAP_SIZE register completes. Each DMA channel contains two shadowed wrap address pointers, SRC_BEG_ADDR and DST_BEG_ADDR, allowing the source and destination wrapping to be independent of each other. Like the SRC_ADDR and DST_ADDR registers, the active SRC/DST_BEG_ADDR registers are loaded from their shadow counterpart at the beginning of a transfer. When the specified number of bursts has occurred, a two part wrap procedure takes place:

- The appropriate active SRC/DST_BEG_ADDR register is incremented by the signed value contained in the SRC/DST_WRAP_STEP register, then
- The new active SRC/DST_BEG_ADDR register is loaded into the active SRC/DST_ADDR register.

Additionally the wrap counter (SRC/DST_WRAP_COUNT) register is reloaded with the SRC/DST_WRAP_SIZE value to setup the next wrap period. This allows the channel to wrap multiple times within a single transfer. Combined with the first bullet above, this allows the channel to address multiple buffers within a single transfer.

The DMA contains both an active and shadow set of the following address pointers. When a DMA transfer begins, the shadow register set is copied to the active working set of registers. This allows you to program the values of the shadow registers for the next transfer while the DMA works with the active set. It also allows you to implement Ping-Pong buffer schemes without disrupting the DMA channel execution.

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 is copied into the active register. 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 will be loaded into the active register at the start of a transfer. On a wrap condition, the active register will be incremented by the signed value in the appropriate SRC/DST_WRAP_STEP register prior to being loaded into the active SRC/DST_ADDR register.

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 a McBSP peripheral, the burst size is limited to 1 since there is no FIFO and the receive or transmit data register must be loaded or copied every word transferred. For RAM the burst size can be up to the maximum allowed by the BURST_SIZE register, which is 32.

Source and Destination Transfer Size (TRANSFER_SIZE): — This specifies the number of bursts to be transferred before 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 reaches 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 McBSP data receive or transmit registers, 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 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.

Three modes are provided to control the way the state machine behaves during the burst loop and the transfer loop:

One Shot Mode (ONESHOT)— If one shot mode is enabled when an interrupt event trigger occurs, the DMA will continue transferring data in bursts until TRANSFER_COUNT is zero. If one shot mode is disabled, then an interrupt event trigger is required for each burst transfer and this will continue until TRANSFER_COUNT is zero.

NOTE: When ONESHOT mode is enabled, the DMA will continuously transfer bursts of data on the given channel until the TRANSFER_COUNT value is zero. This could potentially hog the bandwidth of a peripheral and cause long CPU stalls to occur. To avoid this, you could configure a CPU timer (or similar) and disable ONESHOT so as to avoid this situation.

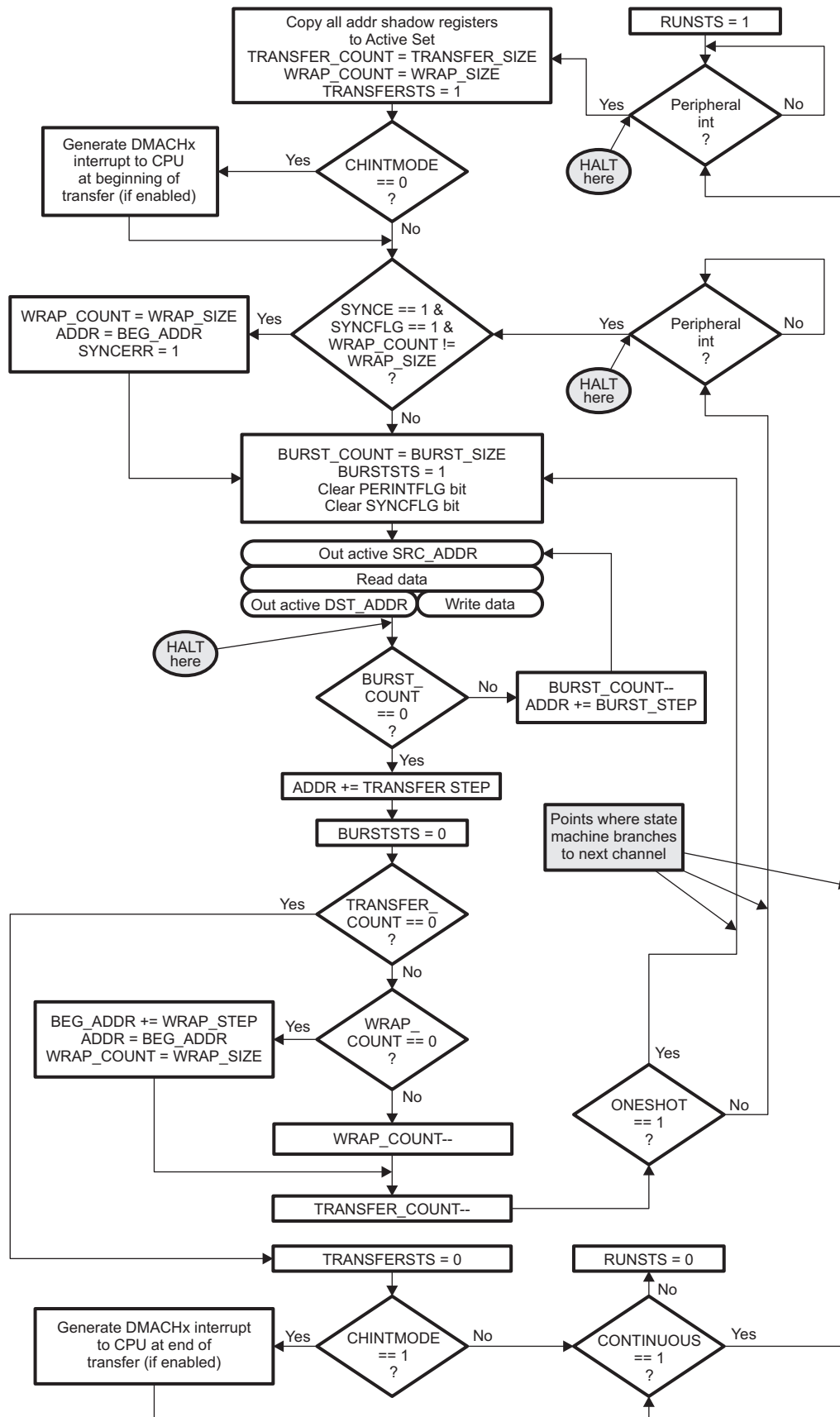
Continuous Mode (CONTINUOUS)— If continuous mode is disabled the RUNSTS bit in the CONTROL register is cleared at the end of the transfer, disabling the DMA channel.

The channel must be re-enabled by setting the RUN bit in the CONTROL register before another transfer can be started on that channel. If the continuous mode is enabled the RUNSTS bit is not cleared at the end of the transfer.

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 4-7](#).

Figure 4-7. DMA State Diagram


The following items are in reference to [Figure 4-7](#).

- 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 ADDR registers are not affected by BEG_ADDR at the start of a transfer. BEG_ADDR only affects the ADDR registers on a wrap or sync error. Following is what happens to each of the ADDR registers when a transfer first starts:
 - BEG_ADDR_SHADOW remains unchanged.
 - ADDR_SHADOW remains unchanged.
 - BEG_ADDR = BEG_ADDR_SHADOW
 - ADDR = ADDR_SHADOW
- The active registers get updated when a wrap occurs. The shadow registers remain unchanged. Specifically:
 - BEG_ADDR_SHADOW remains unchanged.
 - ADDR_SHADOW remains unchanged.
 - BEG_ADDR += WRAP_STEP
 - ADDR = BEG_ADDR
- The active registers get updated when a sync error occurs. The shadow registers remain unchanged. Specifically:
 - BEG_ADDR_SHADOW remains unchanged.
 - ADDR_SHADOW remains unchanged.
 - BEG_ADDR remains unchanged.
 - ADDR = BEG_ADDR

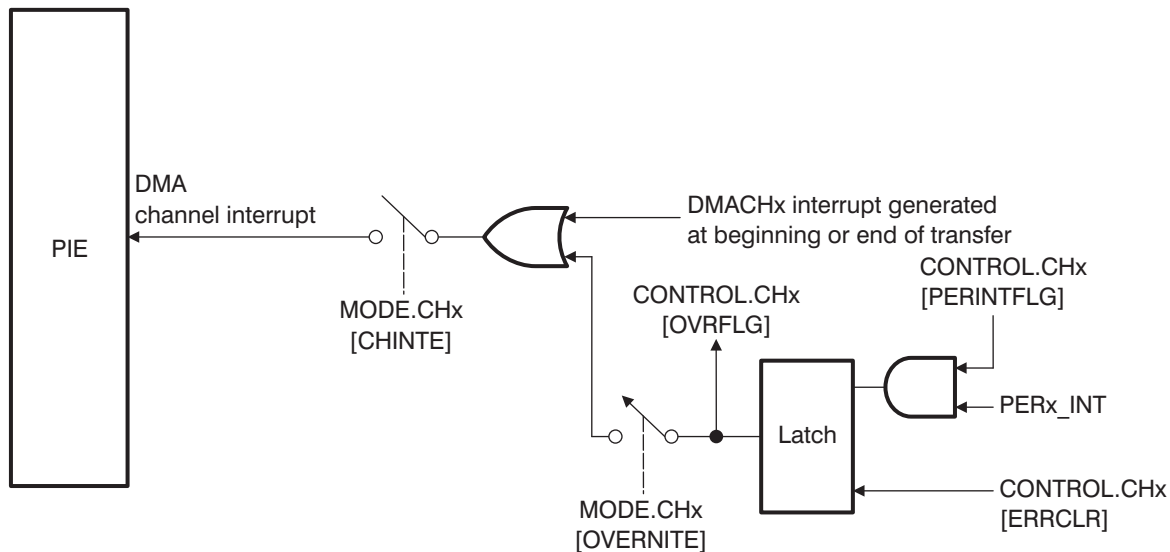
Probably the easiest way to remember all this is that:

- 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.

4.7 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 4-8](#). If the overrun interrupt is enabled then the channel interrupt will be generated to the PIE module.

Figure 4-8. Overrun Detection Logic



4.8 Register Descriptions

The complete DMA register set is shown in [Table 4-2](#).

Table 4-2. DMA Register Summary⁽¹⁾

Address	Acronym	Description	Section
DMA Control, Mode and Status Registers			
0x1000	DMACTRL	DMA Control Register	Section 4.8.1
0x1001	DEBUGCTRL	Debug Control Register	Section 4.8.2
0x1002	REVISION	Peripheral Revision Register	Section 4.8.3
0x1003	Reserved	Reserved	
0x1004	PRIORITYCTRL1	Priority Control Register 1	Section 4.8.4
0x1005	Reserved	Reserved	
0x1006	PRIORITYSTAT	Priority Status Register	Table 4-7
0x1007 0x101F	Reserved	Reserved	
DMA Channel 1 Registers			
0x1020	MODE	Mode Register	Section 4.8.6
0x1021	CONTROL	Control Register	Section 4.8.7
0x1022	BURST_SIZE	Burst Size Register	Section 4.8.8
0x1023	BURST_COUNT	Burst Count Register	Section 4.8.9
0x1024	SRC_BURST_STEP	Source Burst Step Size Register	Section 4.8.10
0x1025	DST_BURST_STEP	Destination Burst Step Size Register	Section 4.8.11
0x1026	TRANSFER_SIZE	Transfer Size Register	Table 4-13
0x1027	TRANSFER_COUNT	Transfer Count Register	Section 4.8.13
0x1028	SRC_TRANSFER_STEP	Source Transfer Step Size Register	Section 4.8.14
0x1029	DST_TRANSFER_STEP	Destination Transfer Step Size Register	Section 4.8.15
0x102A	SRC_WRAP_SIZE	Source Wrap Size Register	Section 4.8.16
0x102B	SRC_WRAP_COUNT	Source Wrap Count Register	Section 4.8.17
0x102C	SRC_WRAP_STEP	Source Wrap Step Size Register	Section 4.8.18
0x102D	DST_WRAP_SIZE	Destination Wrap Size Register	Section 4.8.16
0x102E	DST_WRAP_COUNT	Destination Wrap Count Register	Section 4.8.17
0x102F	DST_WRAP_STEP	Destination Wrap Step Size Register	Section 4.8.18
0x1030	SRC_BEG_ADDR_SHADOW	Shadow Source Begin and Current Address Pointer Registers	Section 4.8.19
0x1032	SRC_ADDR_SHADOW		Section 4.8.19
0x1034	SRC_BEG_ADDR	Active Source Begin and Current Address Pointer Registers	Section 4.8.20
0x1036	SRC_ADDR		Section 4.8.20
0x1038	DST_BEG_ADDR_SHADOW	Shadow Destination Begin and Current Address Pointer Registers	Section 4.8.21
0x103A	DST_ADDR_SHADOW		Section 4.8.21
0x103C	DST_BEG_ADDR	Active Destination Begin and Current Address Pointer Registers	Section 4.8.22
0x103E	DST_ADDR		Section 4.8.22
0x103F	Reserved	Reserved	
DMA Channel 2 Registers			
0x1040 0x105F	Same as above		
DMA Channel 3 Registers			
0x1060 0x107F	Same as above		
DMA Channel 4 Registers			

⁽¹⁾ All DMA register writes are EALLOW protected.

Table 4-2. DMA Register Summary⁽¹⁾ (continued)

Address	Acronym	Description	Section
0x1080 0x109F	Same as above		
DMA Channel 5 Registers			
0x10A0 0x10BF	Same as above		
DMA Channel 6 Registers			
0x10C0 0x10DF	Same as above		

4.8.1 DMA Control Register (DMACTRL) — EALLOW Protected

The DMA control register (DMACTRL) is shown in [Figure 4-9](#) and described in [Table 4-3](#).

Figure 4-9. DMA Control Register (DMACTRL)

15	14	13	12	11	10	9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
			Reserved			PRIORITY RESET	HARD RESET
R-0						R0/S-0	R0/S-0

LEGEND: R0/S = Read 0/Set; R = Read only; -n = value after reset

Table 4-3. DMA Control Register (DMACTRL) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved		Reserved
1	PRIORITYRESET	0	<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, the CH1 burst is completed and then any lower priority channel burst is also completed (if CH1 interrupted in the middle of a burst), before the state machine is reset.</p> <p>In case CH1 is high priority, the <i>state machine</i> restarts from CH2 (or the next highest enabled channel).</p>
0	HARDRESET	0	<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 <i>soft</i> 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 at least a one cycle delay (that is, a NOP instruction) after writing to this bit should be introduced before attempting an access to any other DMA register.</p>

4.8.2 Debug Control Register (DEBUGCTRL) — EALLOW Protected

The debug control register (DEBUGCTRL) is shown in [Figure 4-10](#) and described in [Table 4-4](#).

Figure 4-10. Debug Control Register (DEBUGCTRL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREE		Reserved													
R/W-0		R-0													

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-4. Debug Control Register (DEBUGCTRL) Field Descriptions

Bit	Field	Value	Description
15	FREE	0 1	Emulation Control Bit: This bit specifies the action when an emulation halt event occurs. DMA runs until the current DMA read-write access is completed and the current status of a DMA is frozen. See the <i>HALT</i> points in Figure 4-7 for possible halt states. DMA is unaffected by emulation suspend (run free)
14-0	Reserved		Reserved

4.8.3 Revision Register (REVISION)

The revision register (REVISION) is shown in [Figure 4-11](#) and described in [Table 4-5](#).

Figure 4-11. Revision Register (REVISION)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE								REV							
R								R							

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-5. Revision Register (REVISION) Field Descriptions

Bit	Field	Value	Description
15-8	TYPE	0x0000	DMA Type Bits. A type change represents a major functional feature difference in a peripheral module. Within a peripheral type, there may be minor differences between devices which do not affect the basic functionality of the module. These device-specific differences are listed in the <i>TMS320x28xx, 28xxx DSP Peripheral Reference Guide</i> (SPRU566). This document describes a Type0 DMA.
7-0	REV	0x0000	DMA Silicon Revision Bits: These bits specify the DMA revision and are changed if any bug fixes are performed. First release

4.8.4 Priority Control Register 1 (PRIORITYCTRL1) — EALLOW Protected

The priority control register 1 (PRIORITYCTRL1) is shown in [Figure 4-12](#) and described in [Table 4-6](#).

Figure 4-12. Priority Control Register 1 (PRIORITYCTRL1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CH1 PRIOR ITY	
R-0														R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-6. Priority Control Register 1 (PRIORITYCTRL1) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		Reserved
0	CH1PRIORITY	0 1	DMA Ch1 Priority: This bit selects whether channel 1 has higher priority or not: Same priority as all other channels Highest priority channel Channel priority can only be changed when all channels are disabled. A priority reset should be performed before restarting channels after changing priority.

4.8.5 Priority Status Register (PRIORITYSTAT)

The priority status register (PRIORITYSTAT) is shown in [Figure 4-13](#) and described in [Table 4-7](#).

Figure 4-13. Priority Status Register (PRIORITYSTAT)

15	14	13	12	11	10	9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	ACTIVESTS_SHADOW			Reserved	ACTIVESTS		
R-0	R-0			R-0	R-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-7. Priority Status Register (PRIORITYSTAT) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		Reserved
6-4	ACTIVESTS_SHADOW	0,0,0 No channel pending 0,0,1 CH 1 0,1,0 CH 2 0,1,1 CH 3 1,0,0 CH 4 1,0,1 CH 5 1,1,0 CH 6	Active Channel Status Shadow Bits: These bits are only useful when CH1 is enabled as a higher priority channel. 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:
3	Reserved		Reserved
2-0	ACTIVESTS	0,0,0 no channel active 0,0,1 CH 1 0,1,0 CH 2 0,1,1 CH 3 1,0,0 CH 4 1,0,1 CH 5 1,1,0 CH 6	Active Channel Status Bits: These bits indicate which channel is currently active or performing a transfer:

4.8.6 Mode Register (MODE) — EALLOW Protected

The mode register (MODE) is shown in [Figure 4-14](#) and described in [Table 4-8](#).

Figure 4-14. Mode Register (MODE)

15	14	13	12	11	10	9	8
CHINTE	DATASIZE	Reserved	CONTINUOUS	ONESHOT	CHINTMODE	PERINTE	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
OVRINTE	Reserved				PERINTSEL		
R/W-0	R-0				R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-8. Mode Register (MODE) Field Descriptions

Bit	Field	Value	Description
15	CHINTE	0 1	Channel Interrupt Enable Bit: This bit enables/disables the respective DMA channel interrupt to the CPU (via the PIE). Interrupt disabled Interrupt enabled
14	DATASIZE	0 1	Data Size Mode Bit: This bit selects if the DMA channel transfers 16-bits or 32-bits of data at a time. 16-bit data transfer size 32-bit data transfer size NOTE: Regardless of the value of this bit all of the registers in the DMA refer to 16-bit words. The only effect this bit causes is whether the databus width is 16 or 32 bits. It is up to you to configure the pointer step increment and size to accommodate 32-bit data transfers. See Section 4.6 for details.
13-12	Reserved		Reserved
11	CONTINUOUS		Continuous Mode Bit: If this bit is set to 1, then DMA re-initializes when TRANSFER_COUNT is zero and waits for the next interrupt event trigger. If this bit is 0, then the DMA stops and clears the RUNSTS bit to 0.
10	ONESHOT		One Shot Mode Bit: If this bit is set to 1, then subsequent burst transfers occur without additional event triggers after the first event trigger. If this bit is 0 then only one burst transfer is performed per event trigger.
9	CHINTMODE	0 1	Channel Interrupt Generation Mode Bit: This bit specifies when the respective DMA channel interrupt should be generated to the CPU (via the PIE). Generate interrupt at beginning of new transfer Generate interrupt at end of transfer.
8	PERINTE	0 1	Peripheral Interrupt Trigger Enable Bit: This bit enables/disables the selected peripheral interrupt trigger to the DMA. Interrupt trigger disabled. Neither the selected peripheral nor software can start a DMA burst. Interrupt trigger enabled.
7	OVRINTE	0 1	Overflow Interrupt Enable: This bit when set to 1 enables the DMA to generate an interrupt when an overflow event is detected. Overflow interrupt disabled Overflow interrupt enabled An overflow interrupt is generated when the PERINTFLG is set and another interrupt event occurs. The PERINTFLG being set indicates a previous peripheral event is latched and has not been serviced by the DMA.
6-5	Reserved		Reserved

Table 4-8. Mode Register (MODE) Field Descriptions (continued)

Bit	Field	Value	Description
4-0	PERINTSEL		The Peripheral Interrupt Source Select Bits: These bits should be set to the channel number (i.e. CH2.MODE.PERINTSEL[4:0] = 2).

4.8.7 Control Register (CONTROL) — EALLOW Protected

The control register (CONTROL) is shown in Figure 4-15 and described in Table 4-9.

Figure 4-15. Control Register (CONTROL)

15	14	13	12	11	10	9	8
Reserved	OVRFLG	RUNSTS	BURSTSTS	TRANSFERSTS	Reserved	Reserved	PERINTFLG
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERRCLR	Reserved	PERINTCLR	PERINTFRC	SOFTRESET	HALT	RUN	
R0/S-0	R-0	R0/S-0	R0/S-0	R0/S-0	R0/S-0	R0/S-0	R0/S-0

LEGEND: R0/S = Read 0/Set; R = Read only; -n = value after reset

Table 4-9. Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
15	Reserved		Reserved
14	OVRFLG	0 1	Overflow Flag Bit: This bit indicates if a peripheral interrupt event trigger is received from the selected peripheral and the PERINTFLG is already set. No overflow event Overflow event The ERRCLR bit can be used to clear the state of this bit to 0. The OVRFLG bit is not affected by the PERINTFRC event.
13	RUNSTS	0 1	Run Status Bit: This bit is set to 1 when the RUN bit is written to with a 1. This indicates the DMA channel is now ready to process peripheral interrupt event triggers. This bit is cleared to 0 when TRANSFER_COUNT reaches zero and CONTINUOUS mode bit is set to 0. This bit is also cleared to 0 when either the HARDRESET bit, the SOFTRESET bit, or the HALT bit is activated. Channel is disabled. Channel is enabled.
12	BURSTSTS	0 1	Burst Status Bit: This bit is set to 1 when a DMA burst transfer begins and the BURST_COUNT is initialized with the BURST_SIZE. This bit is cleared to zero when BURST_COUNT reaches zero. This bit is also cleared to 0 when either the HARDRESET or the SOFTRESET bit is activated. No burst activity The DMA is currently servicing or suspending a burst transfer from this channel.
11	TRANSFERSTS	0 1	Transfer Status Bit: This bit is set to 1 when a DMA transfer begins and the address registers are copied to the shadow set and the TRANSFER_COUNT is initialized with the TRANSFER_SIZE. This bit is cleared to zero when TRANSFER_COUNT reaches zero. This bit is also cleared to 0 when either the HARDRESET or the SOFTRESET bit is activated. No transfer activity The channel is currently in the middle of a transfer regardless of whether a burst of data is actively being transferred or not.
10-9	Reserved		Reserved
8	PERINTFLG	0 1	Peripheral Interrupt Trigger Flag Bit: This bit indicates if a peripheral interrupt event trigger has occurred. This flag is automatically cleared when the first burst transfer begins. No interrupt event trigger Interrupt event trigger The PERINTFRC bit can be used to set the state of this bit to 1 and force a software DMA event. The PERINTCLR bit can be used to clear the state of this bit to 0.
7	ERRCLR	0	Error Clear Bit: Writing a 1 to this bit will clear any latched sync error event and clear the SYNCERR bit. This bit will also clear the OVRFLG bit. This bit would normally be used when initializing the DMA for the first time or if an overflow condition is detected. If an ADCSYNC error event or overflow event occurs at the same time as writing to this bit, the ADC or overrun has priority and the SYNCERR or OVRFLG bit is set.
6-5	Reserved		Reserved

Table 4-9. Control Register (CONTROL) Field Descriptions (continued)

Bit	Field	Value	Description
4	PERINTCLR	0	Peripheral Interrupt Clear Bit: Writing a 1 to this bit clears any latched peripheral interrupt event and clears the PERINTFLG bit. This bit would normally be used when initializing the DMA for the first time. If a peripheral event occurs at the same time as writing to this bit, the peripheral has priority and the PERINTFLG bit is set.
3	PERINTFRC	0	Peripheral Interrupt Force Bit: Writing a 1 to this bit latches a peripheral interrupt event trigger and sets the PERINTFLG bit. If the PERINTE bit is set, this bit can be used like a software force for a DMA burst transfer.
2	SOFTRESET	0	Channel Soft Reset Bit: Writing a 1 to this bit completes current read-write access and places the channel into a default state as follows: RUNSTS = 0 TRANSFERSTS = 0 BURSTSTS = 0 BURST_COUNT = 0 TRANSFER_COUNT = 0 SRC_WRAP_COUNT = 0 DST_WRAP_COUNT = 0 This is a <i>soft</i> reset that basically allows the DMA to complete the current read-write access and then places the DMA channel into the default reset state.
1	HALT	0	Channel Halt Bit: Writing a 1 to this bit halts the DMA at the current state and any current read-write access is completed. See Figure 4-7 for the various positions the state machine can be at when HALTED. The RUNSTS bit is set to 0. To take the device out of HALT, the RUN bit needs to be activated.
0	RUN	0	Channel Run Bit: Writing a 1 to this bit starts the DMA channel. The RUNSTS bit is set to 1. This bit is also used to take the device out of HALT. The RUN bit is typically used to start the DMA running after you have configured the DMA. It will then wait for the first interrupt event (PERINTFLG == 1) to start operation. The RUN bit can also be used to take the DMA channel out of a HALT condition. See Figure 4-7 for the various positions the state machine can be at when HALTED.

4.8.8 Burst Size Register (BURST_SIZE) — EALLOW Protected

The burst size register (BURST_SIZE) is shown in [Figure 4-16](#) and described in [Table 4-10](#).

Figure 4-16. Burst Size Register (BURST_SIZE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										BURSTSIZE					
R-0										R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-10. Burst Size Register (BURST_SIZE) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved		Reserved
4-0	BURSTSIZE	0	Transfer 1 word in a burst
		1	Transfer 2 words in a burst
	
		31	Transfer 32 words in a burst

4.8.9 BURST_COUNT Register

The burst count register (BURST_COUNT) is shown in [Figure 4-17](#) and described in [Table 4-11](#).

Figure 4-17. Burst Size Register (BURST_COUNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										BURSTCOUNT					
R-0										R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-11. Burst Count Register (BURST_COUNT) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved		Reserved
4-0	BURSTCOUNT	0	0 word left in a burst
		1	1 word left in a burst
		2	2 words left in a burst
	
		31	31 words left in a burst
			The above values represent the state of the counter at the HALT conditions.

4.8.10 Source Burst Step Register Size (SRC_BURST_STEP) — EALLOW Protected

The source burst step size register (SRC_BURST_STEP) is shown in [Figure 4-18](#) and described in [Table 4-12](#).

Figure 4-18. Source Burst Step Size Register (SRC_BURST_STEP)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRCBURSTSTEP															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-12. Source Burst Step Size Register (SRC_BURST_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	SRCBURSTSTEP		These bits specify the source address post-increment/decrement step size while processing a burst of data:
		0x0FFF	Add 4095 to address
	
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFFE	Sub 2 from address
	
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

4.8.11 Destination Burst Step Register Size (DST_BURST_STEP) — EALLOW Protected

The destination burst step register size (DST_BURST_STEP) is shown in [Figure 4-19](#) and described in [Table 4-13](#).

Figure 4-19. Destination Burst Step Register Size (DST_BURST_STEP)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTBURSTSTEP															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-13. Destination Burst Step Register Size (DST_BURST_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	DSTBURSTSTEP		These bits specify the destination address post-increment/decrement step size while processing a burst of data:
		0x0FFF	Add 4095 to address
	
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFFE	Sub 2 from address
	
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

4.8.12 Transfer Size Register (TRANSFER_SIZE) — EALLOW Protected

The transfer size register (TRANSFER_SIZE) is shown in [Figure 4-20](#) and described in [Table 4-14](#).

Figure 4-20. Transfer Size Register (TRANSFER_SIZE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSFERSIZE															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-14. Transfer Size Register (TRANSFER_SIZE) Field Descriptions

Bit	Field	Value	Description
15-0	TRANSFERSIZE		These bits specify the number of bursts to transfer:
		0x0000	Transfer 1 burst
		0x0001	Transfer 2 bursts
		0x0002	Transfer 3 bursts
	
		0xFFFF	Transfer 65536 bursts

4.8.13 Transfer Count Register (TRANSFER_COUNT)

The transfer count register (TRANSFER_COUNT) is shown in [Figure 4-21](#) and described in [Table 4-15](#).

Figure 4-21. Transfer Count Register (TRANSFER_COUNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSFERCOUNT															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-15. Transfer Count Register (TRANSFER_COUNT) Field Descriptions

Bit	Field	Value	Description
15-0	TRANSFERCOUNT		These bits specify the current transfer counter value:
		0x0000	0 bursts left to transfer
		0x0001	1 burst left to transfer
		0x0002	2 bursts left to transfer
	
		0xFFFF	65535 bursts left to transfer
			The above values represent the state of the counter at the HALT conditions.

4.8.14 Source Transfer Step Size Register (SRC_TRANSFER_STEP) — EALLOW Protected

The source transfer step size register (SRC_TRANSFER_STEP) is shown in [Figure 4-22](#) and described in [Table 4-16](#).

Figure 4-22. Source Transfer Step Size Register (SRC_TRANSFER_STEP)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRCTRANSFERSTEP															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-16. Source Transfer Step Size Register (SRC_TRANSFER_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	SRCTRANSFERSTEP		These bits specify the source address pointer post-increment/decrement step size after processing a burst of data:
		0x0FFF	Add 4095 to address
	
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
	
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

4.8.15 Destination Transfer Step Size Register (DST_TRANSFER_STEP) — EALLOW Protected

The destination transfer step size register (DST_TRANSFER_STEP) is shown in [Figure 4-23](#) and described in [Table 4-17](#).

Figure 4-23. Destination Transfer Step Size Register (DST_TRANSFER_STEP)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSTTRANSFERSTEP															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-17. Destination Transfer Step Size Register (DST_TRANSFER_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	DSTTRANSFERSTEP		These bits specify the destination address pointer post-increment/decrement step size after processing a burst of data:
		0x0FFF	Add 4095 to address
	
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
	
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

4.8.16 Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) — EALLOW protected

The source/destination wrap size register is shown in [Figure 4-24](#) and described in [Table 4-18](#).

Figure 4-24. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRAPSIZE															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-18. Source/Destination Wrap Size Register (SRC/DST_WRAP_SIZE) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPSIZE		These bits specify the number of bursts to transfer before wrapping back to begin address pointer:
		0x0000	Wrap after 1 burst
		0x0001	Wrap after 2 bursts
		0x0002	Wrap after 3 bursts
	
		0xFFFF	Wrap after 65536 bursts
			To <i>disable</i> the wrap function, set the WRAPSIZE bit field to a number larger than the TRANSFERSIZE bit field.

4.8.17 Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)

The source/destination wrap count register (SCR/DST_WRAP_COUNT) is shown in [Figure 4-25](#) and described in [Table 4-19](#).

Figure 4-25. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRAPCOUNT															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-19. Source/Destination Wrap Count Register (SCR/DST_WRAP_COUNT) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPCOUNT		These bits indicate the current wrap counter value:
		0x0000	Wrap complete
		0x0001	1 burst left
		0x0002	2 burst left
	
		0xFFFF	65535 burst left
			The above values represent the state of the counter at the HALT conditions.

4.8.18 Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) — EALLOW Protected

The source/destination wrap step size register (SRC/DST_WRAP_STEP) are shown in [Figure 4-26](#) and described in [Table 4-20](#).

Figure 4-26. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRAPSTEP															
R/W-0															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-20. Source/Destination Wrap Step Size Registers (SRC/DST_WRAP_STEP) Field Descriptions

Bit	Field	Value	Description
15-0	WRAPSTEP		These bits specify the source begin address pointer post-increment/decrement step size after wrap counter expires:
		0x0FFF	Add 4095 to address
	
		0x0002	Add 2 to address
		0x0001	Add 1 to address
		0x0000	No address change
		0xFFFF	Sub 1 from address
		0xFFFE	Sub 2 from address
	
		0xF000	Sub 4096 from address
			Only values from -4096 to 4095 are valid.

4.8.19 Shadow Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) — All EALLOW Protected

The shadow source begin and current address pointer registers (SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) are shown in [Figure 4-27](#) and described in [Table 4-21](#).

**Figure 4-27. Shadow Source Begin and Current Address Pointer Registers
(SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW)**

31	30	29	28	27	26	25	24	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-0																															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4-21. Shadow Source Begin and Current Address Pointer Registers
(SRC_BEG_ADDR_SHADOW/DST_BEG_ADDR_SHADOW) Field Descriptions**

Bit	Field	Value	Description
31-0	BEGADDR		32-bit address value

4.8.20 Active Source Begin and Current Address Pointer Registers (SRC_BEG_ADDR/DST_BEG_ADDR)

The active source begin and current address pointer registers (SRC_BEG_ADDR/DST_BEG_ADDR) are shown in [Figure 4-28](#) and described in [Table 4-22](#).

**Figure 4-28. Active Source Begin and Current Address Pointer Registers
(SRC_BEG_ADDR/DST_BEG_ADDR)**

31	30	29	28	27	26	25	24	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-0																															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4-22. Active Source Begin and Current Address Pointer Registers
(SRC_BEG_ADDR/DST_BEG_ADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	BEGADDR		32-bit address value

4.8.21 Shadow Destination Begin and Current Address Pointer Registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) — All EALLOW Protected

The shadow destination begin and current address pointer registers (SRC_ADDR_SHADOW/DST_ADDR_SHADOW) are shown in [Figure 4-29](#) and described in [Table 4-23](#).

**Figure 4-29. Shadow Destination Begin and Current Address Pointer Registers
(SRC_ADDR_SHADOW/DST_ADDR_SHADOW)**

31	30	29	28	27	26	25	24	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																															
RW-0																															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4-23. Shadow Destination Begin and Current Address Pointer Registers
(SRC_ADDR_SHADOW/DST_ADDR_SHADOW) Field Descriptions**

Bit	Field	Value	Description
31-0	ADDR		32-bit address value

4.8.22 Active Destination Begin and Current Address Pointer Registers (SRC_ADDR/DST_ADDR)

The active destination begin and current address pointer registers (SRC_ADDR/DST_ADDR) are shown in [Figure 4-30](#) and described in [Table 4-24](#).

**Figure 4-30. Active Destination Begin and Current Address Pointer Registers
(SRC_ADDR/DST_ADDR)**

31	30	29	28	27	26	25	24	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-0																															

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4-24. Active Destination Begin and Current Address Pointer Registers
(SRC_ADDR/DST_ADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	ADDR		32-bit address value

Control Law Accelerator (CLA)

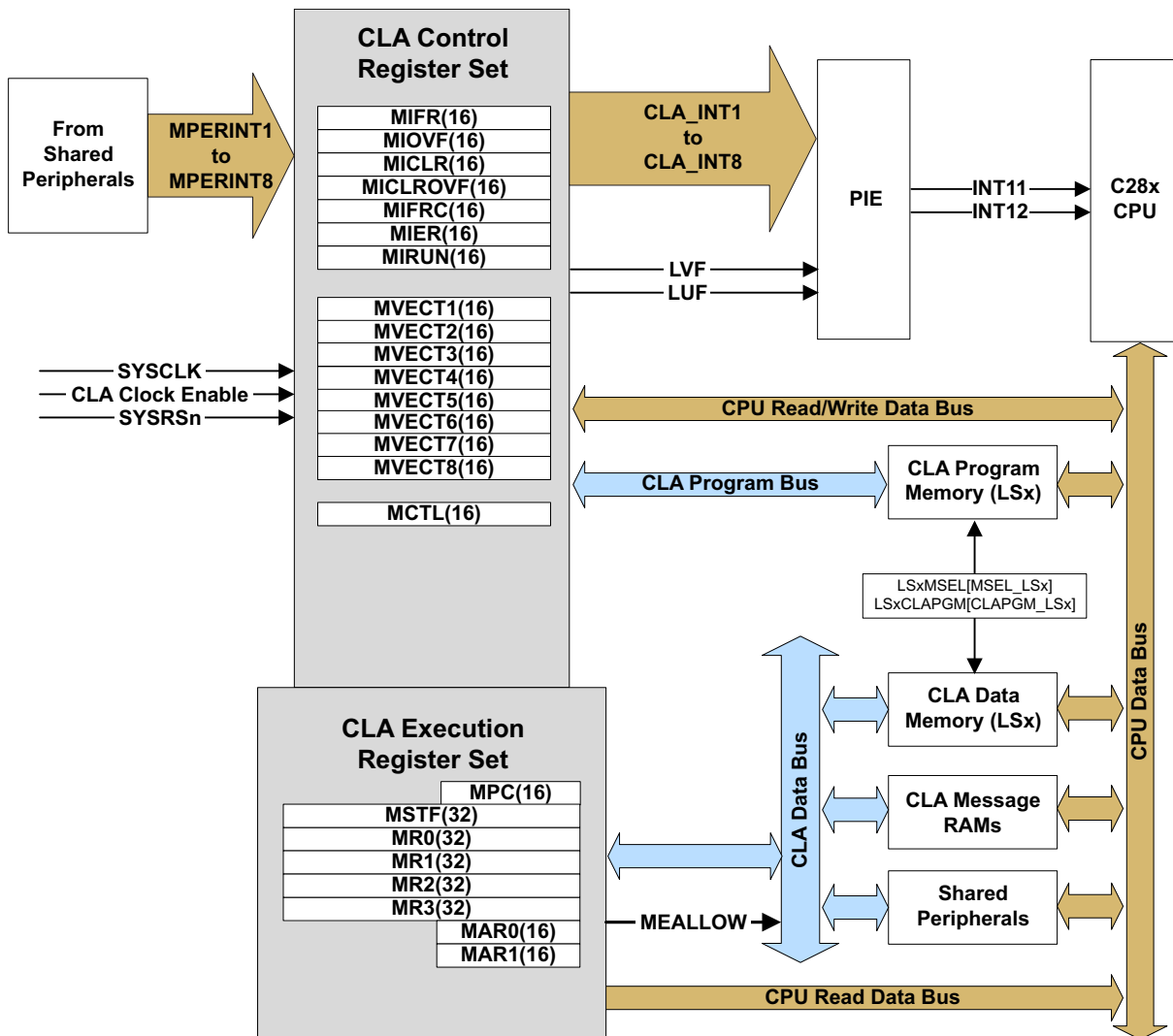
The control law accelerator (CLA) Type-1 is an independent, fully-programmable, 32-bit floating-point math processor that brings concurrent control-loop execution to the C28x family. The low interrupt-latency of the CLA allows it to read ADC samples "just-in-time." This significantly reduces the ADC 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
5.1 Control Law Accelerator (CLA) Overview.....	609
5.2 CLA Interface	611
5.3 CLA and CPU Arbitration	615
5.4 CLA Configuration and Debug	617
5.5 Pipeline	621
5.6 Instruction Set.....	626
5.7 Registers	741

5.1 Control Law Accelerator (CLA) Overview

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. The following is a list of major features of the CLA.

- 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 and program data bus
 - Data address bus, data read bus and data write bus
 - 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.
 - 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 through to 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.
- 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.
- Memory and Shared Peripherals:
 - Two dedicated message RAMs for communication between the CLA and the main CPU.
 - The C28x CPU can map CLA program and data memory to the main CPU space or CLA space.
 - The CLA, on reset, is the secondary master for all peripherals which can have either the CLA or DMA as their secondary master.

Figure 5-1. CLA Block Diagram


5.2 CLA Interface

This chapter describes how the C28x main CPU can interface to the CLA and vice versa.

5.2.1 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.

- **CLA Program Memory**

The CLA program can be loaded to any of the local shared memories (LSxRAM) on the core that it is tied to. At reset, all memory blocks are mapped to the master CPU. While mapped to the CPU space, the CPU can copy the CLA program code into the memory block(s). During debug, the block(s) can also be loaded directly by Code Composer Studio™.

Once the memory is initialized with CLA code, the master 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 an LSx memory is configured as CLA program memory, only debug accesses are allowed on cycles that the CLA is not fetching a new instruction. A detailed explanation of the memory configurations and access arbitration (CPU/CLA/DEBUG) process can be found in [Section 2.11](#).

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 naturally occurs.

- **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 master CPU memory space, whereby the master 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 master 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 an LSx memory is configured as a CLA data memory, the CLA read/write access 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 registers. A detailed explanation of the memory configurations and access arbitration (CPU/CLA/DEBUG) process can be found in [Section 2.11](#).

The CLA RAMs and registers are protected by the security module. Refer to the *DCSM* chapter of this manual for more details on the security scheme.

- **CLA Shared Message RAMs**

There are two small memory blocks for data sharing and communication between the CLA and the master CPU on each CPU subsystem. The message RAMs are always mapped to both CPU and CLA memory spaces and are protected by the DCSM. The message RAMs allow data accesses only; no program fetches can be performed.

- **CLA to CPU Message RAM**

The CLA can use this block to pass data to the master CPU. This block is both readable and writable by the CLA. This block is also readable by the master CPU but writes by the master CPU are ignored.

- **CPU to CLA Message RAM**

The master CPU can use this block to pass data and messages to the CLA. This message RAM is both readable and writable by the master CPU. The CLA can perform reads but writes by the CLA are ignored.

5.2.2 CLA Memory Bus

The CLA has dedicated bus architecture similar to that of the C28x CPU where there is a program read, data read, and data write bus. 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 a 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.

5.2.3 Shared Peripherals and EALLOW Protection

For a given CPU subsystem, the CLA and DMA share secondary access to some peripherals. The secondary ownership of the bus is determined by the CpuSysRegs.SECMSEL[VBUS32_x] bit. If it is set to 0, the CLA is the secondary owner. If it is set to 1, the DMA is the secondary owner. By default, at reset, the CLA is given the secondary ownership of the bus and, therefore, can access all the peripherals connected to it.

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 main CPU status register 1 (ST1) indicates the state of protection for the main 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 main 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 5.5](#).

5.2.4 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 knows where a task begins by the content of the associated interrupt vector (MVECT1 to MVECT8) and the end is indicated by the MSTOP instruction.

The CLA supports eight tasks. Task 1 has the highest priority and task 8 has the lowest priority. A task can be requested by a peripheral interrupt or by software:

- **Peripheral interrupt trigger**

Each task can be triggered by up to 256 interrupt sources. The trigger for each task is defined by writing an appropriate value to the DmaClasrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field. Each of the possible 256 values specifies an interrupt source from a specific peripheral on the shared bus. The configuration options are listed in [Table 5-1](#).

Table 5-1. Configuration Options

Select Value (8-bit)	CLA Trigger Source
0	Software Trigger
1	ADCAINT1
2	ADCAINT2
3	ADCAINT3
4	ADCAINT4
5	ADCAEVT
6	ADCBINT1
7	ADCBINT2
8	ADCBINT3
9	ADCBINT4
10	ADCB EVT
11	ADCCINT1
12	ADCCINT2
13	ADCCINT3
14	ADCCINT4
15	ADCCEVT
16	ADCDINT1
17	ADCDINT2
18	ADCDINT3
19	ADCDINT4
20	ADCDEV T
28:21	Reserved
29	XINT1
30	XINT2
31	XINT3
32	XINT4
33	XINT5
34	Reserved
35	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
44	EPWM9INT
45	EPWM10INT
46	EPWM11INT
47	EPWM12INT
67:48	Reserved
68	TINT0
69	TINT1
70	TINT2
71	MXEVTA
72	MREVTA
73	MXEVTB
74	MREVTB
75	ECAP1INT
76	ECAP2INT
77	ECAP3INT
78	ECAP4INT
79	ECAP5INT
80	ECAP6INT
81	Reserved
82	Reserved
83	EQEP1INT
84	EQEP2INT
85	EQEP3INT
86	Reserved
87	Reserved
88	Reserved
94:89	Reserved
95	SD1INT
96	SD2INT
106:97	Reserved
107	UPP1INT
108	Reserved
109	SPITXINTA
110	SPIRXINTA
111	SPITXINTB
112	SPIRXINTB
113	SPITXINTC
114	SPIRXINTC
255:115	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**

Tasks can also be started by the main CPU software 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.

The CLA has its own fetch mechanism and can run and execute a task independent of the main CPU. Only one task is serviced at a time; there is no nesting of tasks. The task currently running is indicated in the MIRUN register. 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 main CPU.

If the CLA is idle (no task is currently running) 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.
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.

Once a task completes the next highest-priority pending task is automatically serviced and this sequence repeats.

5.2.5 CLA Software Interrupt to CPU

The CLA can issue a software interrupt to the C28x CPU (on the same subsystem) at any point in the code through the use of the CLA1SOFTINTEN and CLA1INTFRC registers. Please see [Section 5.7](#) 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

5.3 CLA and CPU Arbitration

Typically, CLA activity is independent of the CPU activity. Under the circumstance where both the CLA and the CPU are attempting to access memory or a peripheral register within the same interface concurrently, an arbitration procedure will occur. This appendix describes this arbitration.

5.3.1 CLA and CPU Arbitration

The Local Shared RAMs can have multiple masters; that is, the CPU, CLA and the DMA. The arbitration is a combination of fixed and round robin schemes; they are covered in detail in the [Section 2.11](#).

5.3.2 CLA Message RAM

Message RAMs consist of two blocks per CPU subsystem

- CPUx.CLA1 to CPUx – CLA to CPU Message RAM
- CPUx to CPUx.CLA1 – CPU to CLA Message RAM

These blocks are for passing data between the CPU and the CLA. 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 CPUx to CPUx.CLA1 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 2.11](#).

The two message RAMs have the following characteristics:

- CPUx.CLA1 to CPUx Message RAM:
The following accesses are allowed:
 - CPUx reads
 - CPUx.CLA1 data reads and writes
 - CPUx debug reads and writes

The following accesses are ignored:

- CPUx writes
- CPUx to CPUx.CLA1 Message RAM:
The following accesses are allowed:
 - CPUx reads and writes
 - CPUx.CLA1 reads
 - CPUx debug reads and writes

The following accesses are ignored:

- CPUx.CLA1 writes

5.4 CLA Configuration and Debug

This section discusses the steps necessary to configure and debug the CLA.

5.4.1 Building a CLA Application

The control law accelerator can be programmed in either CLA assembly code using the instructions described in [Section 5.6](#) 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). Concurrently the main CPU can perform other tasks.

The CLA type 1 requires Codegen V6.2.4 or later with the following switch: --cla_support=cla1.

5.4.2 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 clock in the PCLKCR3 register.**

PCLKCR3 register is defined in the *System Control and Interrupts* 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.

- **Select the task interrupt sources**

For each task select the interrupt source in the CLA1TASKSRCSELx register. If a task is going to be generated by software, select no interrupt.

- **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.

- **Map CLA data RAM(s) 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 access 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 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 that 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 (ePWM, ADC, and others) that will generate an interrupt to the CLA and be serviced by a CLA task.

The CLA is now ready to service interrupts and the message RAMs can be used to pass data between the CPU and the CLA. Typically mapping of the CLA program and data RAMs occurs only during the initialization process. If after some time you want to re-map these memories back to CPU space then disable interrupts and make sure all tasks have completed by checking the MIRUN register.

5.4.3 Debugging CLA Code

Debugging the CLA code is a simple process that occurs independently of the main CPU.

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 MDEBUGSTOP instruction in the generated assembly does not violate any of the pipeline restrictions.

2. Enable CLA breakpoints

First, enable the CLA breakpoints in the debugger. In Code Composer Studio, this is done by connecting the core from the debug perspective. Breakpoints are disabled when the core is disconnected.

3. Start the task

There are three ways to start the task:

- The peripheral can assert an interrupt
- The main CPU can execute an IACK instruction, or
- You 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, you can single-step the CLA code one cycle at a time. 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 a 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. If desired, disable CLA breakpoints

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.

5.4.4 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 5.4.5](#).

5.4.5 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.

5.5 Pipeline

This section describes the CLA pipeline stages and presents cases where pipeline alignment must be considered.

5.5.1 Pipeline Overview

The CLA pipeline is very similar to the C28x pipeline. The pipeline has eight stages:

- **Fetch 1 (F1)**
During the F1 stage the program read address is placed on the CLA program address bus.
- **Fetch 2 (F2)**
During the F2 stage the instruction is read using the CLA program data bus.
- **Decode 1 (D1)**
During D1 the instruction is decoded.
- **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.
- **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.
- **Read 2 (R2)**
Read the data value using the CLA data read data bus.
- **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.
- **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.

5.5.2 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 5-2](#). 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 5-3](#).

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 5-2. 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 5-3. 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	
						I5	I4	
							I5	

- 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.

Example 5-2. Code Fragment for Loading MAR0 or MAR1

```

; 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 Will use the new value of MAR0 (20)
....

```

5.5.2.1 ADC Early Interrupt to CLA Response

The ADC offers the option to generate an early interrupt pulse when the ADC begins conversion. This option is selected by setting the ADCCTL1[INTPULSEPOS] bit as documented in the Analog-to-Digital Converter and Comparator section in this manual. If this option is used to start a CLA task then the CLA will be able to read the result as soon as the conversion completes and the ADC result register updates. This just-in-time sampling along with the low interrupt response of the CLA enable faster system response and higher frequency control loops.

The timing for the ADC conversion is shown in the ADC Reference Guide timing diagrams. 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 $\frac{1}{4}$ SYSCLK, it would take 10.5 ADCCLK (42 SYSCLK) cycles to complete a conversion. If using the ADC in 16-bit mode at the same ADCCLK, it would take 29.5 ADCCLK (118 SYSCLK) cycles, and so on.

From a CLA perspective, the pipeline activity is shown in Table 5-4 an n-cycle (SYSCLK) conversion. The nth-2 instruction is in the R2 phase just in time to read the result register. While the previous n-3 instructions in the task (I1 to In-3) 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 5-4. ADC to CLA Early Interrupt Response

ADC Activity	CLA Activity	F1	F2	D1	D2	R1	R2	E	W
Sample									
Sample									
...									
Sample									
Conversion (1)	Interrupt Received								
Conversion (2)	Task Startup								
Conversion (3)	Task Startup								
Conversion (4)	I1	I1							
Conversion (5)	I2	I2	I1						
Conversion (6)	I3	I3	I2	I1					
Conversion (7)		
Conversion (n-6)	I(n-6)	I(n-6)							
Conversion (n-5)	I(n-5)	I(n-5)	I(n-6)						
Conversion (n-4)	I(n-4)	I(n-4)	I(n-5)	I(n-6)					
Conversion (n-3)	I(n-3)	I(n-3)	I(n-4)	I(n-5)	I(n-6)				
Conversion (n-2)	I(n-2) Read ADC RESULT	I(n-2)	I(n-3)	I(n-4)	I(n-5)	I(n-6)			
Conversion (n-1)			I(n-2)	I(n-3)	I(n-4)	I(n-5)	I(n-6)		
Conversion (n)				I(n-2)	I(n-3)	I(n-4)	I(n-5)		
Conversion Complete					I(n-2)	I(n-3)	I(n-4)		
RESULT Latched						I(n-2)	I(n-3)		
RESULT Available							I(n-2)		

5.5.3 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

```

; MADD32 || MMOV32 instruction: 32-bit floating-point add with parallel move
; MADD32 is a 1 cycle operation
; MMOV32 is a 1 cycle operation
      MADD32    MR0, MR1, #2    ; MR0 = MR1 + 2,

```

Example 5-3. Math Operation with Parallel Load (continued)

```

|| 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

```

5.6 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.

5.6.1 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 5-5](#).

Table 5-5. 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 5-6. 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 Section 5.5
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).

5.6.2 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 5-7](#).

Table 5-7. Addressing Modes

Addressing Mode	'addr' Opcode Field Encode ⁽¹⁾	Description
@dir	0000	Direct Addressing Mode Example 1: MMOV32 MR1, @_VarA Example 2: MMOV32 MR1, @_EPwm1Regs.CMPA.all In this case the 'mmmm mmmm mmmm mmmm' 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. For example @_VarA will populate the address of the variable VarA. and @_EPwm1Regs.CMPA.all will populate the address of the CMPA register.
*MAR0[#imm16]++	0001	MAR0 Indirect Addressing with 16-bit Immediate Post Increment MAR1 Indirect Addressing with 16-bit Immediate Post Increment 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 Example 1: MMOV32 MR0, *MAR0[2]++ Example 2: MMOV32 MR1, *MAR1[-2]++ For a post increment of 0 the assembler will accept both *MAR0 and *MAR0[0]++. The 'mmmm mmmm mmmm mmmm' 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. If addition of the 16-bit immediate causes overflow, then the value will wrap around on a 16-bit boundary.
*MAR1[#imm16]++	0010	
*MAR0+[#imm16]	0101	MAR0 Offset Addressing with 16-bit Immediate Offset MAR1 Offset Addressing with 16-bit Immediate Offset addr = MAR0 Add the offset #imm16 (or MAR1) + #imm16to address stored in MAR0(MAR1) to access the desired memory the base location Example 1: MMOV32 MR0, *MAR0+[2] Example 1: MMOV32 MR1, *MAR1+[-2] The 'mmmm mmmm mmmm mmmm' 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. If the addition of the 16-bit immediate causes overflow, the value will wrap around on a 16-bit boundary.
*MAR1+[#imm16]	0110	

⁽¹⁾ Values not shown are reserved.

Encoding for the shift fields in the MASR32, MLSR32 and MLSL32 instructions is shown in [Table 5-8](#)

Table 5-8. Shift Field Encoding

Shift Value	'shift' Opcode Field Encode
1	0000
2	0001
3	0010
....
32	1111

Table 5-9 shows the condition field encoding for conditional instructions such as MNEGF, MSWAPF, MBCNDD, MCCNDD, and MRCNDD

Table 5-9. Condition Field Encoding

Encode ⁽¹⁾	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 ⁽²⁾	Unconditional with flag modification	None

⁽¹⁾ Values not shown are reserved.

⁽²⁾ 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.

5.6.3 Instructions

The instructions are listed alphabetically, preceded by a summary.

Table 5-10. General Instructions

Title	Page
MMABSF32 MRa, MRb — 32-Bit Floating-Point Absolute Value.....	632
MADD32 MRa, MRb, MRc — 32-Bit Integer Add.....	633
MADDF32 MRa, #16FHi, MRb — 32-Bit Floating-Point Addition.....	634
MADDF32 MRa, MRb, #16FHi — 32-Bit Floating-Point Addition.....	635
MADDF32 MRa, MRb, MRc — 32-Bit Floating-Point Addition.....	637
MADDF32 MRd, MRe, MRf MMOV32 mem32, MRa — 32-Bit Floating-Point Addition with Parallel Move.....	638
MADDF32 MRd, MRe, MRf MMOV32 MRa, mem32 — 32-Bit Floating-Point Addition with Parallel Move.....	639
MAND32 MRa, MRb, MRc — Bitwise AND.....	641
MASR32 MRa, #SHIFT — Arithmetic Shift Right.....	642
MBCNDD 16BitDest {, CNDF} — Branch Conditional Delayed.....	643
MCCNDD 16BitDest {, CNDF} — Call Conditional Delayed.....	648
MCMP32 MRa, MRb — 32-Bit Integer Compare for Equal, Less Than or Greater Than.....	652
MCMPF32 MRa, MRb — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than.....	653
MCMPF32 MRa, #16FHi — 32-Bit Floating-Point Compare for Equal, Less Than or Greater Than.....	654
MDEBUGSTOP — Debug Stop Task.....	656
MEALLOW — Enable CLA Write Access to EALLOW Protected Registers.....	657
MEDIS — Disable CLA Write Access to EALLOW Protected Registers.....	658
MEINVF32 MRa, MRb — 32-Bit Floating-Point Reciprocal Approximation.....	659
MEISQRTF32 MRa, MRb — 32-Bit Floating-Point Square-Root Reciprocal Approximation.....	660
MF32TOI16 MRa, MRb — Convert 32-Bit Floating-Point Value to 16-Bit Integer.....	661
MF32TOI16R MRa, MRb — Convert 32-Bit Floating-Point Value to 16-Bit Integer and Round.....	662
MF32TOI32 MRa, MRb — Convert 32-Bit Floating-Point Value to 32-Bit Integer.....	663
MF32TOUI16 MRa, MRb — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer.....	664
MF32TOUI16R MRa, MRb — Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round.....	665
MF32TOUI32 MRa, MRb — Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer.....	666
MFRACF32 MRa, MRb — Fractional Portion of a 32-Bit Floating-Point Value.....	667
MI16TOF32 MRa, MRb — Convert 16-Bit Integer to 32-Bit Floating-Point Value.....	668
MI16TOF32 MRa, mem16 — Convert 16-Bit Integer to 32-Bit Floating-Point Value.....	669
MI32TOF32 MRa, mem32 — Convert 32-Bit Integer to 32-Bit Floating-Point Value.....	670
MI32TOF32 MRa, MRb — Convert 32-Bit Integer to 32-Bit Floating-Point Value.....	671
MLSL32 MRa, #SHIFT — Logical Shift Left.....	672
MLSR32 MRa, #SHIFT — Logical Shift Right.....	673
MMACF32 MR3, MR2, MRd, MRe, MRf MMOV32 MRa, mem32 — 32-Bit Floating-Point Multiply and Accumulate with Parallel Move.....	674
MMAXF32 MRa, MRb — 32-Bit Floating-Point Maximum.....	677
MMAXF32 MRa, #16FHi — 32-Bit Floating-Point Maximum.....	679
MMINF32 MRa, MRb — 32-Bit Floating-Point Minimum.....	680
MMINF32 MRa, #16FHi — 32-Bit Floating-Point Minimum.....	682
MMOV16 MARx, MRa, #16I — Load the Auxiliary Register with MRa + 16-bit Immediate Value.....	683
MMOV16 MARx, mem16 — Load MAR1 with 16-bit Value.....	686
MMOV16 mem16, MARx — Move 16-Bit Auxiliary Register Contents to Memory.....	688
MMOV16 mem16, MRa — Move 16-Bit Floating-Point Register Contents to Memory.....	689
MMOV32 mem32, MRa — Move 32-Bit Floating-Point Register Contents to Memory.....	691
MMOV32 mem32, MSTF — Move 32-Bit MSTF Register to Memory.....	692
MMOV32 MRa, mem32 {, CNDF} — Conditional 32-Bit Move.....	693
MMOV32 MRa, MRb {, CNDF} — Conditional 32-Bit Move.....	695

Table 5-10. General Instructions (continued)

MMOV32 MSTF, mem32 — Move 32-Bit Value from Memory to the MSTF Register	697
MMOVD32 MRa, mem32 — Move 32-Bit Value from Memory with Data Copy	698
MMOV32 MRa, #32F — Load the 32-Bits of a 32-Bit Floating-Point Register	699
MMOVI16 MARx, #16I — Load the Auxiliary Register with the 16-Bit Immediate Value	700
MMOVI32 MRa, #32FHex — Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate	701
MMOVIZ MRa, #16FHi — Load the Upper 16-Bits of a 32-Bit Floating-Point Register	702
MMOVZ16 MRa, mem16 — Load MRx With 16-bit Value	703
MMOVXI MRa, #16FLoHex — Move Immediate to the Low 16-Bits of a Floating-Point Register	704
MMPYF32 MRa, MRb, MRc — 32-Bit Floating-Point Multiply	705
MMPYF32 MRa, #16FHi, MRb — 32-Bit Floating-Point Multiply	706
MMPYF32 MRa, MRb, #16FHi — 32-Bit Floating-Point Multiply	708
MMPYF32 MRa, MRb, MRc MADDF32 MRd, MRe, MRf — 32-Bit Floating-Point Multiply with Parallel Add	710
MMPYF32 MRd, MRe, MRf MMOV32 MRa, mem32 — 32-Bit Floating-Point Multiply with Parallel Move	712
MMPYF32 MRd, MRe, MRf MMOV32 mem32, MRa — 32-Bit Floating-Point Multiply with Parallel Move	714
MMPYF32 MRa, MRb, MRc MSUBF32 MRd, MRe, MRf — 32-Bit Floating-Point Multiply with Parallel Subtract	715
MNEGF32 MRa, MRb{, CNDF} — Conditional Negation	716
MNOP — No Operation	718
MOR32 MRa, MRb, MRc — Bitwise OR	719
MRCNDD {CNDF} — Return Conditional Delayed	720
MSETFLG FLAG, VALUE — Set or Clear Selected Floating-Point Status Flags	724
MSTOP — Stop Task	725
MSUB32 MRa, MRb, MRc — 32-Bit Integer Subtraction	727
MSUBF32 MRa, MRb, MRc — 32-Bit Floating-Point Subtraction	728
MSUBF32 MRa, #16FHi, MRb — 32-Bit Floating-Point Subtraction	729
MSUBF32 MRd, MRe, MRf MMOV32 MRa, mem32 — 32-Bit Floating-Point Subtraction with Parallel Move	730
MSUBF32 MRd, MRe, MRf MMOV32 mem32, MRa — 32-Bit Floating-Point Subtraction with Parallel Move	731
MSWAPF MRa, MRb {, CNDF} — Conditional Swap	732
MTESTTF CNDF — Test MSTF Register Flag Condition	734
MUI16TOF32 MRa, mem16 — Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value	736
MUI16TOF32 MRa, MRb — Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value	737
MUI32TOF32 MRa, mem32 — Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value	738
MUI32TOF32 MRa, MRb — Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value	739
MXOR32 MRa, MRb, MRc — Bitwise Exclusive Or	740

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 (0xFFFFFFF4)
    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
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: 0111 0111 1100 bbaa
```

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:
    MMIOVI16    MAR1, #_X          ; Start address
    MUI16TOF32  MR0, @_len         ; Length of the array
    MNOP
    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
```

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, MRc, MRf || MMOV32 MRa, mem32](#)
[MADDF32 MRd, MRc, MRf || MMOV32 mem32, MRa](#)
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRc, 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:
    MOV32 MR0,@M1      ; Load MR0 with M1
    MOV32 MR1,@X1      ; Load MR1 with X1
    MPYF32 MR1,MR1,MR0 ; Multiply M1*X1
    | | MOV32 MR0,@B1    ; and in parallel load MR0 with B1
    MADDF32 MR1,MR1,MR0 ; Add M*X1 to B1 and store in MR1
    MOV32 @Y1,MR1      ; Store the result
    MSTOP              ; end of task
```

See also

[MADDF32 MRa, #16FHi, MRb](#)
[MADDF32 MRa, MRb, #16FHi](#)
[MADDF32 MRd, MRc, MRf || MOV32 MRa, mem32](#)
[MADDF32 MRd, MRc, MRf || MOV32 mem32, MRa](#)
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRc, 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: mmm mmm mmm mmm
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
;
_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
    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: mmm mmm mmm mmm
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, MRc, MRf || MMOV32 mem32, MRa](#)
[MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRc, 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:
    MOV32 MR0, @_m2 ; MR0 = 32 (0x00000020)
    MOV32 MR1, @_x2 ; MR1 = 64 (0x00000040)
    MOV32 MR2, @_b2 ; MR2 = -128 (0xFFFFFFF80)
    MASR32 MR0, #1 ; MR0 = 16 (0x00000010)
    MASR32 MR1, #2 ; MR1 = 16 (0x00000010)
    MASR32 MR2, #3 ; MR2 = -16 (0xFFFFFFF0)
    MOV32 @_m2, MR0 ; store results
    MOV32 @_x2, MR1
    MOV32 @_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 "0xFFFF" 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 ⁽¹⁾	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 ⁽²⁾	Unconditional with flag modification	None

⁽¹⁾ Values not shown are reserved.

⁽²⁾ 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 5-11](#) for each branch 6 instruction slots are executed; three before the branch instruction (I2-I4) and three after the branch instruction (I5-I7). 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 5-11](#) and [Table 5-12](#), 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 5-11. 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 5-12. 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:
    MOV32 MR0, @State
    MCMPPF32 MR0, #0.1           ; Affects flags for 1st MBCNDD (A)
    MCMPPF32 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
    MOV32 MR1, @RampState        ; Always executed
    MOVXI MR2, #RAMPMASK         ; Always executed
    MOR32 MR1, MR2               ; Always executed
    MOV32 @RampState, MR1        ; Execute if (A) branch not taken
    MSTOP                        ; end of task if (A) branch not taken

Skip1:
    MOV32 MR3, @SteadyState
    MOVXI MR2, #STEADYMASK
    MOR32 MR3, MR2
    MBCNDD Skip2, NTF            ; (B) if State != .01, go to Skip2
    MOV32 MR1, @CoastState       ; Always executed
    MOVXI MR2, #COASTMASK        ; Always executed
    MOR32 MR1, MR2               ; Always executed
    MOV32 @CoastState, MR1       ; Execute if (B) branch not taken
    MSTOP                        ; end of task if (B) branch not taken

Skip2:
    MOV32 @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 "0xFFFF" 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 ⁽³⁾	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 ⁽⁴⁾	Unconditional with flag modification	None

⁽³⁾ Values not shown are reserved.

⁽⁴⁾ 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 5-13](#), 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 5-13](#) and [Table 5-14](#), 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 proceeding 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 5-13. 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 5-14. 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 ⁽¹⁾	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	

⁽¹⁾ 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](#)

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](#).

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 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:
    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
        MCMPPF32 MR2, MR1      ; Compare MR2 with MR1
        MSWAPF MR1, MR2, GT    ; MR1 = MAX(MR1, MR2)
        MADDF32 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](#)

MDEBUSTOP

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 MDEBUSTOP instruction is used to halt a task so that it can be debugged. That is, MDEBUSTOP is the CLA breakpoint. If CLA breakpoints are not enabled, the MDEBUSTOP 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 MDEBUSTOP 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](#)

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"
...
_Cla1Task1:
...
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"
...
_Cla1Task1:
...
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)
    MPYF32 MR3, MR2, MR1   ; MR3 = Ye*Den
    MSUBF32 MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MPYF32 MR2, MR2, MR3   ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MPYF32 MR3, MR2, MR1   ; MR3 = Ye*Den
    || MMOV32 MR0, @_Num    ; MR0 = Num
    MSUBF32 MR3, #2.0, MR3 ; MR3 = 2.0 - Ye*Den
    MPYF32 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
    MPYF32 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

```
MMOVF32    MR2, #11204005.0    ; MR2 = 11204005.0 (0x4B2AF5A5)
MF32TOI32   MR3, MR2            ; MR3 = MF32TOI32(MR2) = 11204005 (0x00AAF5A5)
MMOVF32    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: mmmn mmmn mmmn mmmn
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, 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:
    MOV32 MR0, @_m2 ; MR0 = 32 (0x00000020)
    MOV32 MR1, @_x2 ; MR1 = 64 (0x00000040)
    MOV32 MR2, @_b2 ; MR2 = -128 (0xFFFFF80)
    MLSL32 MR0, #1 ; MR0 = 64 (0x00000040)
    MLSL32 MR1, #2 ; MR1 = 256 (0x00000100)
    MLSL32 MR2, #3 ; MR2 = -1024 (0xFFFFFC00)
    MOV32 @_m2, MR0 ; Store results
    MOV32 @_x2, MR1
    MOV32 @_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: mmmmm mmmmm mmmmm mmmmm
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:
    MMOVI16 MAR0, #_X          ; MAR0 points to X array
    MMOVI16 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
    | 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

```

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 (0xBF000000)
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:
    MOVII16 MAR1, #_X ; Start address
    MUI16TOF32 MR0, @_len ; Length of the array
    MNOP ; delay for MAR1 load
    MNOP ; delay for MAR1 load
    MOV32 MR1, *MAR1[2]++ ; MR1 = X0
LOOP
    MOV32 MR2, *MAR1[2]++ ; MR2 = next element
    MMAXF32 MR1, MR2 ; MR1 = MAX(MR1, MR2)
    MADDF32 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
    MOV32 @_Result, MR1 ; Always executed
    MNOP ; Always executed
    MNOP ; Always executed
    MSTOP ; End of task
```

See also

[MCMPPF32 MRa, MRb](#)
[MCMPPF32 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
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

[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 5-15. 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
    ML32 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 5-16. 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
    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
    MCMFPF32   MR0, #NUM_DATA_POINTS.0         ;I33 Compare count to max
    MF32TOUI16 MR0, MR0                        ;I34 Convert count to Uint16
    MNOP
    MMOVZ16    MR2, @_AdcaResultRegs.ADCRESULT1 ;I35 Wait till I36 to read result
    MMOV16     *MAR1, MR2                      ;I36 Read ADCRESULT1
    MBCNDD     _RestartCount, GEQ              ; Store ADCRESULT1
    MMOVIZ     MR1, #0.0                       ; If count >= NUM_DATA_POINTS
    MNOP
    MNOP
    MMOV16     @_ConversionCount, MR0          ; Always executed: MR1=0
    MSTOP
                                           ; If branch not taken
                                           ; 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: mmm mmm mmm mmm (Opcode for MMOV16 mem16, MAR0)
MSW: 0111 0110 1000 addr

LSW: mmm mmm mmm mmm (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: mmmmmmmmmmmmmmmmmmm
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
    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

[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: mmmmmmmmmmmmmmmmm
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
    MOPYF32   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
    MOPYF32   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
    MOPYF32   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: mmm mmm mmm mmm
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: mmm mmm mmm mmm
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 ⁽¹⁾	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 ⁽²⁾	Unconditional with flag modification	None

⁽¹⁾ Values not shown are reserved.

⁽²⁾ 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
;
_ClalTask5:
    MMOV32    MR0, @_A
    MMOV32    MR1, @_B
    MCMFPF32  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 ⁽³⁾	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 ⁽⁴⁾	Unconditional with flag modification	None

⁽³⁾ Values not shown are reserved.

⁽⁴⁾ 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: mmm mmm mmm mmm
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](#)

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 5-17. 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: 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 (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

```
MOVI32 MR1, #0x40400000 ; MR1 = 0x40400000
                        ; Assembler converts this instruction as
                        ; MMOVIZ MR1, #0x4040

MOVI32 MR2, #0x00000000 ; MR2 = 0x00000000
                        ; Assembler converts this instruction as
                        ; MMOVIZ MR2, #0x0

MOVI32 MR3, #0x40004001 ; MR3 = 0x40004001
                        ; Assembler converts this instruction as
                        ; MMOVIZ MR3, #0x4000
                        ; MMOVXI MR3, #0x4001

MOVI32 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 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](#)
[MMOVI32 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: mmm mmm mmm mmm
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
    MEINV32   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, MRe, MRf](#)
[MMPYF32 MRd, MRe, MRf || MMOV32 MRa, mem32](#)
[MMPYF32 MRd, MRe, MRf || MMOV32 mem32, MRa](#)
[MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf](#)
[MMACF32 MR3, MR2, MRd, MRe, 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.
MRc	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 0 0 0 b a 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, 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
MSW: 0 1 1 1 0 1 1 1 1 0 0 0 b a 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:
    MMOVI16    MAR0, #_X                ; MAR0 points to X array
    MMOVI16    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[2]++          ; In parallel MR0 = X4
    MMOV32     MR1, *MAR1[2]++          ; 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

[MMACF32 MR3, MR2, MRd, MRc, 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: mmm mmm mmm mmm
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
;
_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
```

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: mmm mmm mmm mmm
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 ddcc 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:
    MMOV32    MR0, @A           ; Load MR0 with A
    MMOV32    MR1, @B           ; Load MR1 with B
    MMPYF32   MR2, MR0, MR1     ; Multiply (A*B)
    || MSUBF32 MR3, MR0, MR1     ; and in parallel Sub (A-B)
    MMOV32    @Y2, MR2          ; Store A*B
    MMOV32    @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 ⁽⁵⁾	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 ⁽⁶⁾	Unconditional with flag modification	None

⁽⁵⁾ Values not shown are reserved.

⁽⁶⁾ 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)
    MPMYF32   MR3, MR2, MR1     ; MR3 = Ye*Den
    MSUBF32   MR3, #2.0, MR3    ; MR3 = 2.0 - Ye*Den
    MPMYF32   MR2, MR2, MR3     ; MR2 = Ye = Ye*(2.0 - Ye*Den)
    MPMYF32   MR3, MR2, MR1     ; MR3 = Ye*Den
    || MMOV32  MR0, @_Num        ; MR0 = Num
    MSUBF32   MR3, #2.0, MR3    ; MR3 = 2.0 - Ye*Den
    MPMYF32   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
    MPMYF32   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
    MUII6TOF32 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
    MCMPIF32   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 ⁽⁷⁾	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 ⁽⁸⁾	Unconditional with flag modification	None

⁽⁷⁾ Values not shown are reserved.

⁽⁸⁾ 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 5-18](#), 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 5-18](#) and [Table 5-19](#), 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 5-18. Pipeline Activity For MRCNDD, Return Not Taken

Instruction	F1	F2	D1	D2	R1	R2	E	W
d4	d4	d3	d2	d1	I7	I6	I5	
d5	d5	d4	d3	d2	d1	I7	I6	
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 5-19. Pipeline Activity For MRCNDD, Return Taken

Instruction	F1	F2	D1	D2	R1	R2	E	W
d4	d4	d3	d2	d1	I7	I6	I5	
d5	d5	d4	d3	d2	d1	I7	I6	
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	
I8	I8	d10	d9	d8	-	d7	d6	
I9	I9	I8	d10	d9	d8	-	d7	
I10	I10	I9	I8	d10	d9	d8	-	
etc....	I10	I9	I8	d10	d9	d8	
....		I10	I9	I8	d10	d9	
....			I10	I9	I8	d10	
					I10	I9	I8	
						I10	I9	
							I10	

Example

;

See also

[MBCNDD #16BitDest, CNDF](#)
[MCCNDD 16BitDest, CNDF](#)
[MMOV32 mem32, MSTF](#)
[MMOV32 MSTF, mem32](#)

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 MSTFLG 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 5-20](#) 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 5-20. 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
_Cla1Task3:
    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](#)

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:
    MOV32    MR0, @_A          ; MR0 = 1 (0x00000001)
    MOV32    MR1, @_B          ; MR1 = 2 (0x00000002)
    MOV32    MR2, @_C          ; MR2 = -7 (0xFFFFFFFF9)
    MSUB32    MR3, MR0, MR1     ; A + B
    MSUB32    MR3, MR3, MR2     ; A + B + C = 6 (0x00000006)
    MOV32    @_y2, MR3         ; Store y2
    MSTOP
```

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: 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
```

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:
    MOV32    MR0, @_x          ; MR0 = X
    MEISQRTF32 MR1, MR0        ; MR1 = Ye = Estimate(1/sqrt(X))
    MOV32    MR1, @_x, EQ       ; if(X == 0.0) Ye = 0.0
    MPYF32   MR3, MR0, #0.5     ; MR3 = X*0.5
    MPYF32   MR2, MR1, MR3      ; MR2 = Ye*X*0.5
    MPYF32   MR2, MR1, MR2      ; MR2 = Ye*Ye*X*0.5
    MSUBF32  MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
    MPYF32   MR1, MR1, MR2      ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MPYF32   MR2, MR1, MR3      ; MR2 = Ye*X*0.5
    MPYF32   MR2, MR1, MR2      ; MR2 = Ye*Ye*X*0.5
    MSUBF32  MR2, #1.5, MR2     ; MR2 = 1.5 - Ye*Ye*X*0.5
    MPYF32   MR1, MR1, MR2      ; MR1 = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    MPYF32   MR0, MR1, MR0      ; MR0 = Y = Ye*X
    MOV32    @_y, MR0          ; Store Y = sqrt(X)
    MSTOP
```

See also

MSUBF32 MRa, MRb, MRc
MSUBF32 MRd, MRe, MRf || MOV32 MRa, mem32
MSUBF32 MRd, MRe, MRf || MOV32 mem32, MRa
MPYF32 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: rrrrr rrrrr rrrrr rrrrr
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: mmm mmm mmm mmm
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 ⁽¹⁾	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 ⁽²⁾	Unconditional with flag modification	None

⁽¹⁾ Values not shown are reserved.

⁽²⁾ 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: MCMPPF32 and MSWAPF can be replaced by MMAXF32
;
_ClalTask1:
    MMOV16    MAR1, #_X          ; Start address
    MUI16TOF32 MR0, @_len        ; Length of the array
    MNOP
    MNOP
    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
    MNOP
    MSTOP
    ; Always executed
    ; 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 ⁽³⁾	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 ⁽⁴⁾	Unconditional with flag modification	None

⁽³⁾ Values not shown are reserved.

⁽⁴⁾ 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:
    MOV32    MR0, @_State
    MCMPPF32 MR0, #0.1      ; Affects flags for 1st MBCNDD (A)
    MCMPPF32 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
    MOV32    MR1, @_RampState ; Always executed
    MOVXI    MR2, #RAMPMASK  ; Always executed
    MOR32    MR1, MR2        ; Always executed
    MOV32    @_RampState, MR1 ; Execute if (A) branch not taken
    MSTOP    ; end of task if (A) branch not taken

_Skip1:
    MOV32    MR3, @_SteadyState
    MOVXI    MR2, #STEADYMASK
    MOR32    MR3, MR2
    MBCNDD   _Skip2, NTF     ; (B) if State != .01, go to Skip2
    MOV32    MR1, @_CoastState ; Always executed
    MOVXI    MR2, #COASTMASK  ; Always executed
    MOR32    MR1, MR2        ; Always executed
    MOV32    @_CoastState, MR1 ; Execute if (B) branch not taken
    MSTOP    ; end of task if (B) branch not taken

_Skip2:
    MOV32    @_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: mmm mmm mmm mmm

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: mmm mmm mmm mmm
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)
    MPPYF32      MR3, MR0, MR1 ; M*X
    MADDF32      MR3, MR2, MR3 ; Y=MX+B = 5.0 (0x40A00000)
    MF32TOUI32   MR3, MR3      ; Y = Uint32(5.0) = 0x00000005
    MMOV32       @_y2, MR3     ; store result
    MSTOP
```

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](#)

5.7 Registers

5.7.1 CLA Base Addresses

Table 5-21. CLA Base Address Table

Device Register	Register Name	Start Address	End Address
Cla1Regs	CLA_REGS	0x0000_1400	0x0000_147F
Cla1SoftIntRegs ⁽¹⁾	CLA_SOFTINT_REGS	0x0000_0CE0	0x0000_0CFF

⁽¹⁾ This register is only accessible from the CLA.

5.7.2 CLA_REGS Registers

Table 5-22 lists the memory-mapped registers for the CLA_REGS. All register offset addresses not listed in Table 5-22 should be considered as reserved locations and the register contents should not be modified.

Table 5-22. CLA_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	MVECT1	Task Interrupt Vector		Go
1h	MVECT2	Task Interrupt Vector		Go
2h	MVECT3	Task Interrupt Vector		Go
3h	MVECT4	Task Interrupt Vector		Go
4h	MVECT5	Task Interrupt Vector		Go
5h	MVECT6	Task Interrupt Vector		Go
6h	MVECT7	Task Interrupt Vector		Go
7h	MVECT8	Task Interrupt Vector		Go
10h	MCTL	Control Register		Go
20h	MIFR	Interrupt Flag Register		Go
21h	MIOVF	Interrupt Overflow Flag Register		Go
22h	MIFRC	Interrupt Force Register		Go
23h	MICLR	Interrupt Flag Clear Register		Go
24h	MICLROVF	Interrupt Overflow Flag Clear Register		Go
25h	MIER	Interrupt Enable Register		Go
26h	MIRUN	Interrupt Run Status Register		Go
28h	_MPC	CLA Program Counter		Go
2Ah	_MAR0	CLA Auxiliary Register 0		Go
2Bh	_MAR1	CLA Auxiliary Register 1		Go
2Eh	_MSTF	CLA Floating-Point Status Register		Go
30h	_MR0	CLA Floating-Point Result Register 0		Go
34h	_MR1	CLA Floating-Point Result Register 1		Go
38h	_MR2	CLA Floating-Point Result Register 2		Go
3Ch	_MR3	CLA Floating-Point Result Register 3		Go

5.7.2.1 MVECT1 Register (Offset = 0h) [reset = 0h]

MVECT1 is shown in [Figure 5-2](#) and described in [Table 5-23](#).

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 5-2. MVECT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-23. 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 MCLA 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>

5.7.2.2 MVECT2 Register (Offset = 1h) [reset = 0h]

MVECT2 is shown in [Figure 5-3](#) and described in [Table 5-24](#).

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 5-3. MVECT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-24. 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 MCLA 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>

5.7.2.3 MVECT3 Register (Offset = 2h) [reset = 0h]

MVECT3 is shown in [Figure 5-4](#) and described in [Table 5-25](#).

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 5-4. MVECT3 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-25. 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 MCLA 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>

5.7.2.4 MVECT4 Register (Offset = 3h) [reset = 0h]

MVECT4 is shown in [Figure 5-5](#) and described in [Table 5-26](#).

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 5-5. MVECT4 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-26. 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 MCLA 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>

5.7.2.5 MVECT5 Register (Offset = 4h) [reset = 0h]

MVECT5 is shown in [Figure 5-6](#) and described in [Table 5-27](#).

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 5-6. MVECT5 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-27. 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 MCLA 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>

5.7.2.6 MVECT6 Register (Offset = 5h) [reset = 0h]

MVECT6 is shown in [Figure 5-7](#) and described in [Table 5-28](#).

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 5-7. MVECT6 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-28. 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 MCLA 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>

5.7.2.7 MVECT7 Register (Offset = 6h) [reset = 0h]

MVECT7 is shown in [Figure 5-8](#) and described in [Table 5-29](#).

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 5-8. MVECT7 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-29. 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 MCLA 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>

5.7.2.8 MVECT8 Register (Offset = 7h) [reset = 0h]

MVECT8 is shown in [Figure 5-9](#) and described in [Table 5-30](#).

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 5-9. MVECT8 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MVECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-30. 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 MCLA 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>

5.7.2.9 MCTL Register (Offset = 10h) [reset = 0h]

MCTL is shown in [Figure 5-10](#) and described in [Table 5-31](#).

Control Register

Figure 5-10. 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/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-31. MCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	IACKE	R/W	0h	<p>IACK enable</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</p> <p>IACK #0x0003 Write a 1 to MIFRC bit 0 and 1 to force task 1 and task 2</p>
1	SOFTRESET	R/W	0h	<p>Soft Reset</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/W	0h	<p>Hard Reset</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>

5.7.2.10 MIFR Register (Offset = 20h) [reset = 0h]

MIFR is shown in [Figure 5-11](#) and described in [Table 5-32](#).

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 5-11. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-32. MIFR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	Task 8 Interrupt Flag 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
6	INT7	R	0h	Task 7 Interrupt Flag 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
5	INT6	R	0h	Task 6 Interrupt Flag 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
4	INT5	R	0h	Task 5 Interrupt Flag 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
3	INT4	R	0h	Task 4 Interrupt Flag 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

Table 5-32. MIFR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	INT3	R	0h	Task 3 Interrupt Flag 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
1	INT2	R	0h	Task 2 Interrupt Flag 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
0	INT1	R	0h	Task 1 Interrupt Flag 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

5.7.2.11 MIOVF Register (Offset = 21h) [reset = 0h]

MIOVF is shown in [Figure 5-12](#) and described in [Table 5-33](#).

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 5-12. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-33. MIOVF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	Task 8 Interrupt Overflow Flag 0h (R/W) = A task 8 interrupt overflow has not occurred (default) 1h (R/W) = A task 8 interrupt overflow has occurred
6	INT7	R	0h	Task 7 Interrupt Overflow Flag 0h (R/W) = A task 7 interrupt overflow has not occurred (default) 1h (R/W) = A task 7 interrupt overflow has occurred
5	INT6	R	0h	Task 6 Interrupt Overflow Flag 0h (R/W) = A task 6 interrupt overflow has not occurred (default) 1h (R/W) = A task 6 interrupt overflow has occurred
4	INT5	R	0h	Task 5 Interrupt Overflow Flag 0h (R/W) = A task 5 interrupt overflow has not occurred (default) 1h (R/W) = A task 5 interrupt overflow has occurred
3	INT4	R	0h	Task 4 Interrupt Overflow Flag 0h (R/W) = A task 4 interrupt overflow has not occurred (default) 1h (R/W) = A task 4 interrupt overflow has occurred
2	INT3	R	0h	Task 3 Interrupt Overflow Flag 0h (R/W) = A task 3 interrupt overflow has not occurred (default) 1h (R/W) = A task 3 interrupt overflow has occurred
1	INT2	R	0h	Task 2 Interrupt Overflow Flag 0h (R/W) = A task 2 interrupt overflow has not occurred (default) 1h (R/W) = A task 2 interrupt overflow has occurred
0	INT1	R	0h	Task 1 Interrupt Overflow Flag 0h (R/W) = A task 1 interrupt overflow has not occurred (default) 1h (R/W) = A task 1 interrupt overflow has occurred

5.7.2.12 MIFRC Register (Offset = 22h) [reset = 0h]

MIFRC is shown in [Figure 5-13](#) and described in [Table 5-34](#).

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 5-13. 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/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-34. MIFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R/W	0h	Task 8 Interrupt Force 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/W	0h	Task 7 Interrupt Force 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/W	0h	Task 6 Interrupt Force 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/W	0h	Task 5 Interrupt Force 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
3	INT4	R/W	0h	Task 4 Interrupt Force 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 4 interrupt
2	INT3	R/W	0h	Task 3 Interrupt Force 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 3 interrupt
1	INT2	R/W	0h	Task 2 Interrupt Force 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 2 interrupt
0	INT1	R/W	0h	Task 1 Interrupt Force 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 1 interrupt

5.7.2.13 MICLR Register (Offset = 23h) [reset = 0h]

MICLR is shown in [Figure 5-14](#) and described in [Table 5-35](#).

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 5-14. 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/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-35. MICLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R/W	0h	Task 8 Interrupt Flag Clear 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/W	0h	Task 7 Interrupt Flag Clear 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/W	0h	Task 6 Interrupt Flag Clear 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/W	0h	Task 5 Interrupt Flag Clear 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
3	INT4	R/W	0h	Task 4 Interrupt Flag Clear 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 4 interrupt flag
2	INT3	R/W	0h	Task 3 Interrupt Flag Clear 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 3 interrupt flag
1	INT2	R/W	0h	Task 2 Interrupt Flag Clear 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 2 interrupt flag
0	INT1	R/W	0h	Task 1 Interrupt Flag Clear 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 1 interrupt flag

5.7.2.14 MICLROVF Register (Offset = 24h) [reset = 0h]

MICLROVF is shown in [Figure 5-15](#) and described in [Table 5-36](#).

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 5-15. 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/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-36. MICLROVF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R/W	0h	Task 8 Interrupt Overflow Flag Clear 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/W	0h	Task 7 Interrupt Overflow Flag Clear 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/W	0h	Task 6 Interrupt Overflow Flag Clear 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/W	0h	Task 5 Interrupt Overflow Flag Clear 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
3	INT4	R/W	0h	Task 4 Interrupt Overflow Flag Clear 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 4 interrupt overflow flag
2	INT3	R/W	0h	Task 3 Interrupt Overflow Flag Clear 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 3 interrupt overflow flag
1	INT2	R/W	0h	Task 2 Interrupt Overflow Flag Clear 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 2 interrupt overflow flag
0	INT1	R/W	0h	Task 1 Interrupt Overflow Flag Clear 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 1 interrupt overflow flag

5.7.2.15 MIER Register (Offset = 25h) [reset = 0h]

MIER is shown in [Figure 5-16](#) and described in [Table 5-37](#).

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 5-16. 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-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-37. MIER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	Task 8 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 8 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 8 interrupt is enabled
6	INT7	R	0h	Task 7 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 7 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 7 interrupt is enabled
5	INT6	R	0h	Task 6 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 6 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 6 interrupt is enabled
4	INT5	R	0h	Task 5 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 5 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 5 interrupt is enabled
3	INT4	R	0h	Task 4 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 4 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 4 interrupt is enabled
2	INT3	R	0h	Task 3 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 3 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 3 interrupt is enabled

Table 5-37. MIER Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	INT2	R	0h	Task 2 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 2 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 2 interrupt is enabled
0	INT1	R	0h	Task 1 Interrupt Enable 0h (R/W) = TASK_INT_DISABLE Task 1 interrupt is disabled (default) 1h (R/W) = TASK_INT_ENABLE Task 1 interrupt is enabled

5.7.2.16 MIRUN Register (Offset = 26h) [reset = 0h]

MIRUN is shown in [Figure 5-17](#) and described in [Table 5-38](#).

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 5-17. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-38. MIRUN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	INT8	R	0h	Task 8 Run Status 0h (R/W) = Task 8 is not executing (default) 1h (R/W) = Task 8 is executing
6	INT7	R	0h	Task 7 Run Status 0h (R/W) = Task 7 is not executing (default) 1h (R/W) = Task 7 is executing
5	INT6	R	0h	Task 6 Run Status 0h (R/W) = Task 6 is not executing (default) 1h (R/W) = Task 6 is executing
4	INT5	R	0h	Task 5 Run Status 0h (R/W) = Task 5 is not executing (default) 1h (R/W) = Task 5 is executing
3	INT4	R	0h	Task 4 Run Status 0h (R/W) = Task 4 is not executing (default) 1h (R/W) = Task 4 is executing
2	INT3	R	0h	Task 3 Run Status 0h (R/W) = Task 3 is not executing (default) 1h (R/W) = Task 3 is executing
1	INT2	R	0h	Task 2 Run Status 0h (R/W) = Task 2 is not executing (default) 1h (R/W) = Task 2 is executing
0	INT1	R	0h	Task 1 Run Status 0h (R/W) = Task 1 is not executing (default) 1h (R/W) = Task 1 is executing

5.7.2.17 _MPC Register (Offset = 28h) [reset = 0h]

_MPC is shown in [Figure 5-18](#) and described in [Table 5-39](#).

CLA Program Counter

Figure 5-18. _MPC Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
_MPC															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-39. _MPC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	_MPC	R/W	0h	Points to the instruction currently in the decode 2 phase of the CLA pipeline. The address range of the CLA with a 16-bit MPC is 64K 16-bit words or 32K CLA instructions.

5.7.2.18 _MAR0 Register (Offset = 2Ah) [reset = 0h]

_MAR0 is shown in [Figure 5-19](#) and described in [Table 5-40](#).

CLA Auxiliary Register 0

Figure 5-19. _MAR0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
_MAR0															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-40. _MAR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	_MAR0	R/W	0h	CLA Auxillary Register 0

5.7.2.19 _MAR1 Register (Offset = 2Bh) [reset = 0h]

_MAR1 is shown in [Figure 5-20](#) and described in [Table 5-41](#).

CLA Auxiliary Register 1

Figure 5-20. _MAR1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
_MAR1															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-41. _MAR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	_MAR1	R/W	0h	CLA Auxillary Register 1

5.7.2.20 _MSTF Register (Offset = 2Eh) [reset = 0h]

_MSTF is shown in [Figure 5-21](#) and described in [Table 5-42](#).

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, MLRS32
- 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 5-21. _MSTF Register

31	30	29	28	27	26	25	24
RESERVED				_RPC			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
_RPC							
R/W-0h							
15	14	13	12	11	10	9	8
_RPC				MEALLOW	RESERVED	RNDF32	RESERVED
R/W-0h				R/W-0h	R-0h	R/W-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	TF	RESERVED		ZF	NF	LUF	LVF
R-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-42. _MSTF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-12	_RPC	R/W	0h	Return program counter The RPC is used to save and restore the MPC address by the MCCNDD and MRCNDD operations
11	MEALLOW	R/W	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. 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/W	0h	Round 32-bit Floating-Point Mode Use the MSETFLG and MMOV32 MSTF, mem32 instructions to change the rounding mode. 0h (R/W) = If this bit is zero, the MMPYF32, MADDF32 and MSUBF32 instructions will round to zero (truncate). 1h (R/W) = If this bit is one, the MMPYF32, MADDF32 and MSUBF32 instructions will round to the nearest even value.
8-7	RESERVED	R	0h	Reserved

Table 5-42. _MSTF Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	TF	R/W	0h	<p>Test Flag</p> <p>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.</p> <p>0h (R/W) = The condition tested with the MTESTTF instruction is false.</p> <p>1h (R/W) = The condition tested with the MTESTTF instruction is true.</p>
5-4	RESERVED	R/W	0h	Reserved
3	ZF	R/W	0h	<p>Zero Flag</p> <p>- Instructions that modify this flag based on the floating-point value stored in the destination register:</p> <p>MMOV32, MMOVD32, MABSF32, MNEGF32</p> <p>- Instructions that modify this flag based on the floating-point result of the operation:</p> <p>MCMPF32, MMAXF32, and MMINF32</p> <p>- Instructions that modify this flag based on the integer result of the operation:</p> <p>MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and MLSL32</p> <p>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag.</p> <p>0h (R/W) = The value is not zero</p> <p>1h (R/W) = The value is zero</p>
2	NF	R/W	0h	<p>Negative Flag</p> <p>- Instructions that modify this flag based on the floating-point value stored in the destination register:</p> <p>MMOV32, MMOVD32, MABSF32, MNEGF32</p> <p>- Instructions that modify this flag based on the floating-point result of the operation:</p> <p>MCMPF32, MMAXF32, and MMINF32</p> <p>- Instructions that modify this flag based on the integer result of the operation:</p> <p>MMOV16, MAND32, MOR32, MXOR32, MCMP32, MASR32, MLSR32 and MLSL32</p> <p>The MSETFLG and MMOV32 MSTF, mem32 instructions can also be used to modify this flag.</p> <p>0h (R/W) = The value is not negative</p> <p>1h (R/W) = The value is negative</p>
1	LUF	R/W	0h	<p>Latched Underflow Flag</p> <p>The following instructions will set this flag to 1 if an underflow 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>0h (R/W) = An underflow condition has not been latched</p> <p>1h (R/W) = An underflow condition has been latched</p>

Table 5-42. _MSTF Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	LVF	R/W	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>0h (R/W) = An overflow condition has not been latched</p> <p>1h (R/W) = An overflow condition has been latched</p>

5.7.2.21 _MR0 Register (Offset = 30h) [reset = 0h]

_MR0 is shown in [Figure 5-22](#) and described in [Table 5-43](#).

CLA Floating-Point Result Register 0

Figure 5-22. _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/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-43. _MR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	CLA Result Register 0

5.7.2.22 _MR1 Register (Offset = 34h) [reset = 0h]

_MR1 is shown in [Figure 5-23](#) and described in [Table 5-44](#).

CLA Floating-Point Result Register 1

Figure 5-23. _MR1 Register

31	30	29	28	27	26	25	24	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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-44. _MR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	CLA Result Register 1

5.7.2.23 _MR2 Register (Offset = 38h) [reset = 0h]

_MR2 is shown in [Figure 5-24](#) and described in [Table 5-45](#).

CLA Floating-Point Result Register 2

Figure 5-24. _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/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-45. _MR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	CLA Result Register 2

5.7.2.24 _MR3 Register (Offset = 3Ch) [reset = 0h]

_MR3 is shown in [Figure 5-25](#) and described in [Table 5-46](#).

CLA Floating-Point Result Register 3

Figure 5-25. _MR3 Register

31	30	29	28	27	26	25	24	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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-46. _MR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	i32	R/W	0h	CLA Result Register 3

5.7.3 CLA_SOFTINT_REGS Registers

[Table 5-47](#) lists the memory-mapped registers for the CLA_SOFTINT_REGS. All register offset addresses not listed in [Table 5-47](#) should be considered as reserved locations and the register contents should not be modified.

Table 5-47. CLA_SOFTINT_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SOFTINTEN	Task Software Interrupt Enable		Go
2h	SOFTINTFRC	Task Software Interrupt Force		Go

5.7.3.1 SOFTINTEN Register (Offset = 0h) [reset = 0h]

SOFTINTEN is shown in [Figure 5-26](#) and described in [Table 5-48](#).

This register is only accessible by the CLA (not the CPU).

Figure 5-26. SOFTINTEN 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
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-48. SOFTINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	TASK8	R/W	0h	Task 8 Software Interrupt Enable 0h (R/W) = Task8 Software Interrupt is disabled. 1h (R/W) = Task8 Software Interrupt is enabled.
6	TASK7	R/W	0h	Task 7 Software Interrupt Enable 0h (R/W) = Task7 Software Interrupt is disabled. 1h (R/W) = Task7 Software Interrupt is enabled.
5	TASK6	R/W	0h	Task 6 Software Interrupt Enable 0h (R/W) = Task6 Software Interrupt is disabled. 1h (R/W) = Task6 Software Interrupt is enabled.
4	TASK5	R/W	0h	Task 5 Software Interrupt Enable 0h (R/W) = Task5 Software Interrupt is disabled. 1h (R/W) = Task5 Software Interrupt is enabled.
3	TASK4	R/W	0h	Task 4 Software Interrupt Enable 0h (R/W) = Task4 Software Interrupt is disabled. 1h (R/W) = Task4 Software Interrupt is enabled.
2	TASK3	R/W	0h	Task 3 Software Interrupt Enable 0h (R/W) = Task3 Software Interrupt is disabled. 1h (R/W) = Task3 Software Interrupt is enabled.
1	TASK2	R/W	0h	Task 2 Software Interrupt Enable 0h (R/W) = Task2 Software Interrupt is disabled. 1h (R/W) = Task2 Software Interrupt is enabled.
0	TASK1	R/W	0h	Task 1 Software Interrupt Enable 0h (R/W) = Task1 Software Interrupt is disabled. 1h (R/W) = Task1 Software Interrupt is enabled.

5.7.3.2 SOFTINTFRC Register (Offset = 2h) [reset = 0h]

SOFTINTFRC is shown in [Figure 5-27](#) and described in [Table 5-49](#).

This register is only accessible by the CLA (not the CPU).

Figure 5-27. SOFTINTFRC 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
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 5-49. SOFTINTFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	TASK8	R/W	0h	Task 8 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task8 Software Interrupt
6	TASK7	R/W	0h	Task 7 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task7 Software Interrupt
5	TASK6	R/W	0h	Task 6 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task6 Software Interrupt
4	TASK5	R/W	0h	Task 5 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task5 Software Interrupt
3	TASK4	R/W	0h	Task 4 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task4 Software Interrupt
2	TASK3	R/W	0h	Task 3 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task3 Software Interrupt
1	TASK2	R/W	0h	Task 2 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task2 Software Interrupt
0	TASK1	R/W	0h	Task 1 Software Interrupt Force 0h (R/W) = No action performed 1h (R/W) = Forces Task1 Software Interrupt

Inter-Processor Communication (IPC)

Topic	Page
6.1 Inter-Processor Communication	775
6.2 Message RAMs.....	776
6.3 IPC Flags and Interrupts.....	776
6.4 IPC Command Registers	776
6.5 Flash Pump and Clock Configuration Semaphores.....	777
6.6 Free-Running Counter.....	778
6.7 IPC Communication Protocol	779
6.8 Registers	780

6.1 Inter-Processor Communication

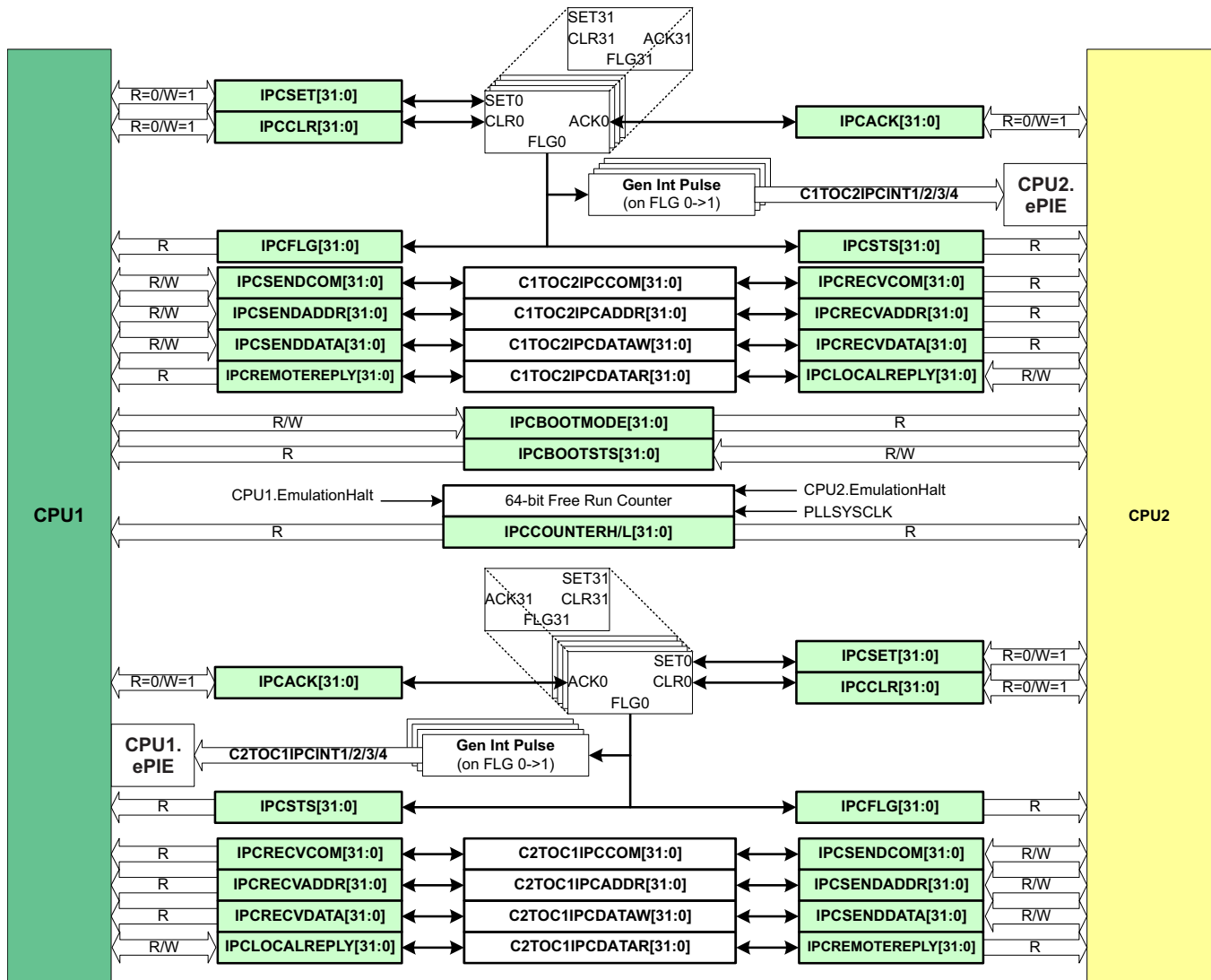
The Inter-Processor Communications (IPC) module allows communication between the two CPU subsystems. This section details the IPC features that each CPU can use to request and share information. The IPC features are:

- Message RAMs
- IPC flags and interrupts
- IPC command registers
- Flash pump semaphore
- Clock configuration semaphore
- Free-running counter

All IPC features are independent of each other, and most do not require any specific data format. There are also two registers for boot mode/status communication. Please refer to the boot ROM chapter for more information on these registers.

Figure 6-1 shows the design structure of the IPC module.

Figure 6-1. IPC Module Architecture



6.2 Message RAMs

There are two dedicated 2kB blocks of message RAM. Each CPU and its DMA have read/write access to one RAM and read-only access to the other RAM, as shown in [Table 6-1](#)

Table 6-1. IPC Message RAM Read/Write Access

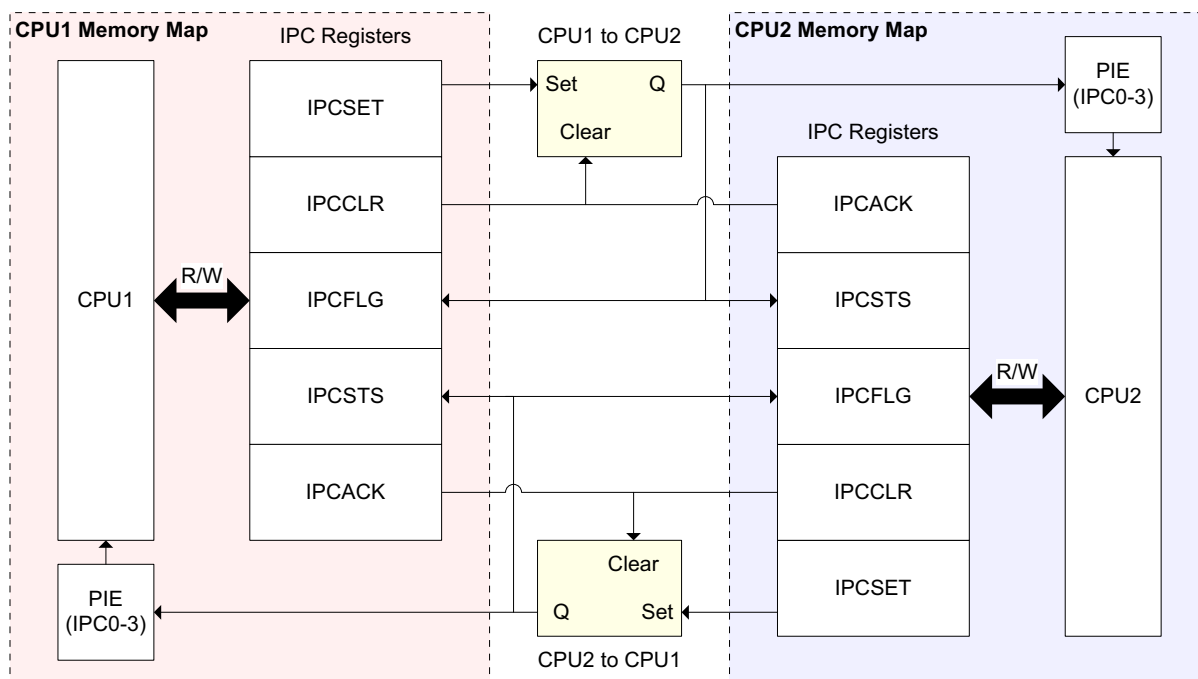
	CPU1	CPU2	CPU1 DMA	CPU2 DMA
CPU1 to CPU2 (1K x 16, address 0x03FC00)	R/W	R	R/W	R
CPU2 to CPU1 (1K x 16, address 0x03F800)	R	R/W	R	R/W

Reading or writing a message RAM does not trigger any events on the remote CPU.

6.3 IPC Flags and Interrupts

There are 32 IPC event signals from CPU1 to CPU2, and vice-versa. These signals can be used for flag-based event polling. Four of them (IPC0 - IPC3) can be configured to generate IPC interrupts on the remote CPU. [Figure 6-2](#) shows the IPC flag messaging and interrupt system.

Figure 6-2. Messaging with IPC Flags and Interrupts



6.4 IPC Command Registers

The IPC command registers provide a simple and flexible way for the two CPUs to exchange more complex messages. Each CPU has eight dedicated registers, four for sending messages and four for receiving messages. The register names were chosen to support a simple command/response protocol, but can be used for any purpose. Only the read/write permissions are determined by hardware; the data format is entirely software-defined.

For sending messages, each CPU has three writable registers and one read-only register. Those same registers are accessible on the remote CPU as three read-only registers and one writable register.

[Table 6-2](#) shows the command registers.

Table 6-2. IPC Command Registers

Local Register Name	Local CPU	Remote CPU	Remote Register Name
IPCSENDCOM	R/W	R	IPCRECVCOM
IPCSENDADDR	R/W	R	IPCRECVADDR
IPCSENDDATA	R/W	R	IPCRECVDATA
IPCREMOTEREPLY	R	R/W	IPCLOCALREPLY

6.5 Flash Pump and Clock Configuration Semaphores

The semaphore registers allow the two CPUs to manage shared system resources.

6.5.1 Flash Pump Semaphore

Each CPU subsystem has its own flash bank, which it can read, program, and erase. Both flash banks share a single charge pump for program and erase operations. Hence, only one CPU can program/erase its flash at any given time. A CPU can read data and execute code from its flash even when the other CPU is programming or erasing. The flash pump semaphore allows one CPU to take control of the pump without being interrupted by the other CPU.

6.5.2 Clock Configuration Semaphore

The two CPUs share a common set of PLL and peripheral clock configuration registers. Since each CPU can have its own mix of peripherals, both may access the configuration registers. The clock configuration semaphore allows one CPU to access the registers without being interrupted by the other CPU.

6.5.3 Semaphore States

Both the flash pump and the clock configuration semaphores allow the same states and state transitions. The semaphore is implemented as a two-bit field in a register with special write protections. The semaphore registers require a key field to be written at the same time as the semaphore bits. See the PUMPREQUEST and CLKSEM register descriptions for details. The possible semaphore states are:

00 or 11	Either CPU may write to the semaphore. CPU1 has control of the resource by default. 00 is the reset state.
01	CPU2 has exclusive control of the resource and exclusive write access to the semaphore.
10	CPU1 has exclusive control of the resource and exclusive write access to the semaphore.

Each CPU is only allowed to take control of the resource for itself. Direct transfer between the 01 and 10 states is not allowed. However, CPU1 may force both semaphores into the default state (00) at any time by putting CPU2 into reset. [Figure 6-3](#) and [Figure 6-4](#) show the allowed states and state transitions.

Figure 6-3. Flash Pump Semaphore State transitions

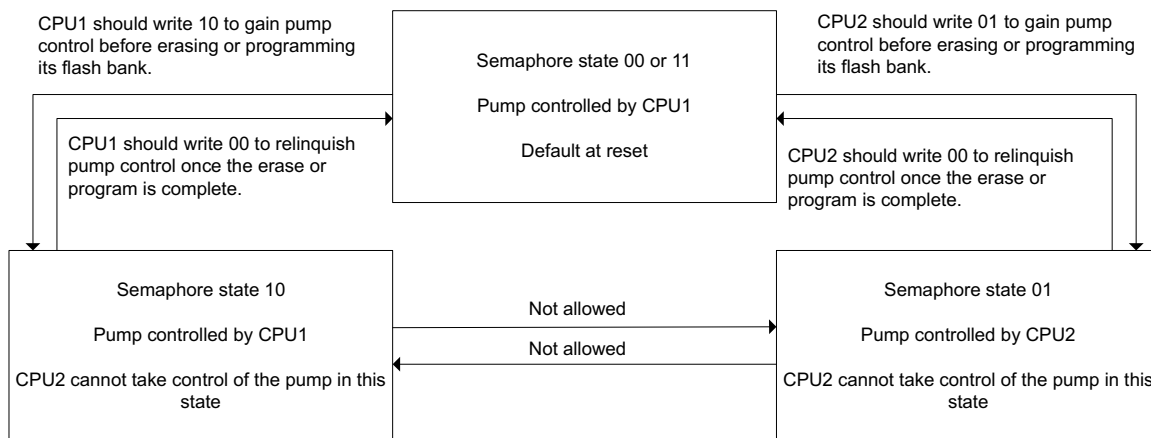
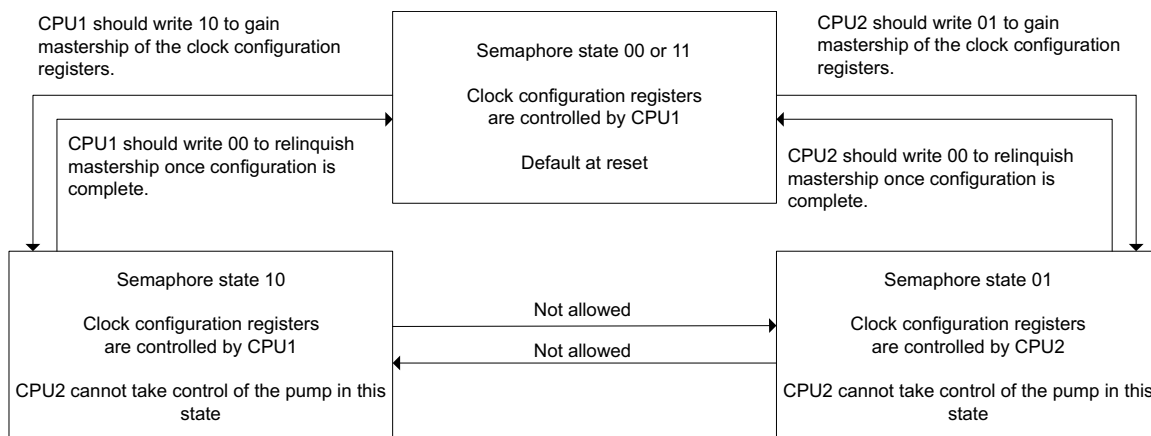


Figure 6-4. Clock Configuration Semaphore State Transitions



6.6 Free-Running Counter

A 64-bit free-running counter is present in the device and can be used to timestamp IPC events between processors. The counter is clocked by PLLSYSCLK and reset by SYSRSn. The counter is implemented as two 32-bit registers, IPCCOUNTERH and IPCCOUNTERL. When IPCCOUNTERL is read, the value of IPCCOUNTERH is saved. A subsequent read to IPCCOUNTERH returns this saved value. This design prevents race conditions due to IPCCOUNTERL overflowing between reads of the two registers.

The free-running counter only stops when both CPUs are in HALT mode. If either core is executing, the counter runs.

6.7 IPC Communication Protocol

This section describes the hardware support options for IPC communication between the two CPUs. These options can be used independently or in combination. All flag definitions and data formats are entirely user-defined.

- The flag system supports event-based communication via interrupts and register polling.
 - CPUx can raise an IPC event by writing to any of the 32 bits of its IPCSET register. This sets the corresponding bits in CPUx's IPCFLG register and CPUy's IPCSTS register.
 - CPUy can signal its response to the event by setting the appropriate bit in its IPCACK register. This clears the corresponding bits in CPUx's IPCFLG register and CPUy's IPCSTS register.
 - If CPUx needs to cancel an event, it can set the appropriate bit in its IPCCLR register. This has the same effect as CPUy writing to IPCACK.
 - Flags 0–3 (set via IPCSET[3:0]) fire interrupts to the remote CPU. The remote CPU must configure its ePIE module properly in order to receive an IPC interrupt. Flags 4–31 (set via IPCSET[31:4]) do not produce interrupts. Multiple flags can be set, acknowledged, and cleared simultaneously.
- The command registers support sending several distinct pieces of information. They are named COM, ADDR, DATA, and REPLY for convenience only and can hold whatever data the application needs.
 - CPUx can write data to its IPCSENDCOM, IPCSENDADDR, and IPCSENDATA registers. CPUy receives these in its IPCRECVCOM, IPCRECVADDR, and IPCRECVDATA registers.
 - CPUy can respond by writing to its IPCLOCALREPLY register. CPUx receives this data in its IPCREMOTEREPLY register.
 - Each CPU can only write to its SEND and LOCALREPLY registers. The RECV and REMOTEREPLY registers are read-only.
- There is an additional pair of command-like registers offered for boot-time IPC or any other convenient use — IPCBOOTMODE and IPCBOOTSTS. Both CPUs can read these registers. CPU1 can only write to IPCBOOTMODE, and CPU2 can only write to IPCBOOTSTS.
- There are two shared memories for passing large amounts of data between the CPUs. Each CPU has a writable memory for sending data and a read-only memory for receiving data.
- Here is an example of how to use these features together. CPUx needs some data from CPUy's LS RAM. The data is at CPUy address 0x9400 and is 0x80 16-bit words long. The protocol could be implemented like this:
 - CPUx writes 0x1 to IPCSENDCOM, defined in software to mean "copy data from address". It writes the address (0x9400) to IPCSENDADDR and the data length (0x80) to IPCSENDATA.
 - CPUx writes to IPCSET[3] and IPCSET[16]. Here, IPC flag 3 is configured to send an interrupt and IPCSET[16] is defined in software to indicate an incoming command. CPUx begins polling for IPCFLG[3] to go low.
 - CPUy receives the interrupt. In the interrupt handler, it checks IPCSTS, finds that flag 16 is set, and runs a command processor.
 - CPUy reads the command (0x1) from IPCRECVCOM, the address (0x9400) from IPCRECVADDR, and the data length (0x80) from IPCRECVDATA. CPUy then copies the LS RAM data to an empty space in its writable shared memory starting at offset 0x210.
 - CPUy writes the shared memory address (0x210) to its IPCLOCALREPLY register. It then writes to IPCACK[16] and IPCACK[3] to clear the flags and indicate completion of the command. CPUy's work is done.
 - CPUx sees IPCFLG[3] go low. It reads IPCREMOTEREPLY to get the shared memory offset of the copied data (0x210).

6.8 Registers

6.8.1 IPC Base Addresses

Table 6-3. IPC Base Addresses

Device Registers	Register Name	Start Address	End Address
IpcRegs (CPU1)	IPC_REGS_CPU1	0x0005_0000	0x0005_0FFF
IpcRegs (CPU2)	IPC_REGS_CPU2	0x0005_0000	0x0005_0FFF

6.8.2 IPC_REGS_CPU1 Registers

Table 6-4 lists the memory-mapped registers for the IPC_REGS_CPU1. All register offset addresses not listed in Table 6-4 should be considered as reserved locations and the register contents should not be modified.

Table 6-4. IPC_REGS_CPU1 Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	IPCACK	IPC incoming flag clear (acknowledge) register		Go
2h	IPCSTS	IPC incoming flag status register		Go
4h	IPCSET	IPC remote flag set register		Go
6h	IPCCLR	IPC remote flag clear register		Go
8h	IPCFLG	IPC remote flag status register		Go
Ch	IPCCOUNTERL	IPC Counter Low Register		Go
Eh	IPCCOUNTERH	IPC Counter High Register		Go
10h	IPSENDCOM	Local to Remote IPC Command Register		Go
12h	IPSENDADDR	Local to Remote IPC Address Register		Go
14h	IPSENDATA	Local to Remote IPC Data Register		Go
16h	IPCREMOTEREPLY	Remote to Local IPC Reply Data Register		Go
18h	IPCRCVCOM	Remote to Local IPC Command Register		Go
1Ah	IPCRCVADDR	Remote to Local IPC Address Register		Go
1Ch	IPCRCVDATA	Remote to Local IPC Data Register		Go
1Eh	IPCLOCALREPLY	Local to Remote IPC Reply Data Register		Go
20h	IPCBOOTSTS	CPU2 to CPU1 IPC Boot Status Register		Go
22h	IPCBOOTMODE	CPU1 to CPU2 IPC Boot Mode Register		Go
24h	PUMPREQUEST	Flash programming semaphore PUMP request register	EALLOW	Go

6.8.2.1 IPCACK Register (Offset = 0h) [reset = 0h]

IPCACK is shown in [Figure 6-23](#) and described in [Table 6-24](#).

IPC incoming flag clear (acknowledge) register

Figure 6-5. IPCACK Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-5. IPCACK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit clears the IPC31 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit clears the IPC30 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit clears the IPC29 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit clears the IPC28 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit clears the IPC27 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
26	IPC26	R=0/W=1	0h	Writing 1 to this bit clears the IPC26 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit clears the IPC25 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit clears the IPC24 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit clears the IPC23 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit clears the IPC22 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

Table 6-5. IPCACK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21	IPC21	R=0/W=1	0h	Writing 1 to this bit clears the IPC21 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit clears the IPC20 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit clears the IPC19 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
18	IPC18	R=0/W=1	0h	Writing 1 to this bit clears the IPC18 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
17	IPC17	R=0/W=1	0h	Writing 1 to this bit clears the IPC17 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit clears the IPC16 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit clears the IPC15 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit clears the IPC14 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit clears the IPC13 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit clears the IPC12 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit clears the IPC11 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit clears the IPC10 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit clears the IPC9 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
8	IPC8	R=0/W=1	0h	Writing 1 to this bit clears the IPC8 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit clears the IPC7 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit clears the IPC6 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit clears the IPC5 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

Table 6-5. IPCACK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	IPC4	R=0/W=1	0h	Writing 1 to this bit clears the IPC4 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit clears the IPC3 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit clears the IPC2 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit clears the IPC1 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
0	IPC0	R=0/W=1	0h	Writing 1 to this bit clears the IPC0 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

6.8.2.2 IPCSTS Register (Offset = 2h) [reset = 0h]

IPCSTS is shown in [Figure 6-24](#) and described in [Table 6-25](#).

IPC incoming flag status register

Figure 6-6. IPCSTS Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-6. IPCSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R	0h	Indicates to the local CPU if the IPC31 event flag was set by the remote CPU. 0: No IPC31 event was set by the remote CPU 1: An IPC31 event was set by the remote CPU
30	IPC30	R	0h	Indicates to the local CPU if the IPC30 event flag was set by the remote CPU. 0: No IPC30 event was set by the remote CPU 1: An IPC30 event was set by the remote CPU
29	IPC29	R	0h	Indicates to the local CPU if the IPC29 event flag was set by the remote CPU. 0: No IPC29 event was set by the remote CPU 1: An IPC29 event was set by the remote CPU
28	IPC28	R	0h	Indicates to the local CPU if the IPC28 event flag was set by the remote CPU. 0: No IPC28 event was set by the remote CPU 1: An IPC28 event was set by the remote CPU
27	IPC27	R	0h	Indicates to the local CPU if the IPC27 event flag was set by the remote CPU. 0: No IPC27 event was set by the remote CPU 1: An IPC27 event was set by the remote CPU
26	IPC26	R	0h	Indicates to the local CPU if the IPC26 event flag was set by the remote CPU. 0: No IPC26 event was set by the remote CPU 1: An IPC26 event was set by the remote CPU
25	IPC25	R	0h	Indicates to the local CPU if the IPC25 event flag was set by the remote CPU. 0: No IPC25 event was set by the remote CPU 1: An IPC25 event was set by the remote CPU

Table 6-6. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	IPC24	R	0h	Indicates to the local CPU if the IPC24 event flag was set by the remote CPU. 0: No IPC24 event was set by the remote CPU 1: An IPC24 event was set by the remote CPU
23	IPC23	R	0h	Indicates to the local CPU if the IPC23 event flag was set by the remote CPU. 0: No IPC23 event was set by the remote CPU 1: An IPC23 event was set by the remote CPU
22	IPC22	R	0h	Indicates to the local CPU if the IPC22 event flag was set by the remote CPU. 0: No IPC22 event was set by the remote CPU 1: An IPC22 event was set by the remote CPU
21	IPC21	R	0h	Indicates to the local CPU if the IPC21 event flag was set by the remote CPU. 0: No IPC21 event was set by the remote CPU 1: An IPC21 event was set by the remote CPU
20	IPC20	R	0h	Indicates to the local CPU if the IPC20 event flag was set by the remote CPU. 0: No IPC20 event was set by the remote CPU 1: An IPC20 event was set by the remote CPU
19	IPC19	R	0h	Indicates to the local CPU if the IPC19 event flag was set by the remote CPU. 0: No IPC19 event was set by the remote CPU 1: An IPC19 event was set by the remote CPU
18	IPC18	R	0h	Indicates to the local CPU if the IPC18 event flag was set by the remote CPU. 0: No IPC18 event was set by the remote CPU 1: An IPC18 event was set by the remote CPU
17	IPC17	R	0h	Indicates to the local CPU if the IPC17 event flag was set by the remote CPU. 0: No IPC17 event was set by the remote CPU 1: An IPC17 event was set by the remote CPU
16	IPC16	R	0h	Indicates to the local CPU if the IPC16 event flag was set by the remote CPU. 0: No IPC16 event was set by the remote CPU 1: An IPC16 event was set by the remote CPU
15	IPC15	R	0h	Indicates to the local CPU if the IPC15 event flag was set by the remote CPU. 0: No IPC15 event was set by the remote CPU 1: An IPC15 event was set by the remote CPU
14	IPC14	R	0h	Indicates to the local CPU if the IPC14 event flag was set by the remote CPU. 0: No IPC14 event was set by the remote CPU 1: An IPC14 event was set by the remote CPU
13	IPC13	R	0h	Indicates to the local CPU if the IPC13 event flag was set by the remote CPU. 0: No IPC13 event was set by the remote CPU 1: An IPC13 event was set by the remote CPU
12	IPC12	R	0h	Indicates to the local CPU if the IPC12 event flag was set by the remote CPU. 0: No IPC12 event was set by the remote CPU 1: An IPC12 event was set by the remote CPU

Table 6-6. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	IPC11	R	0h	Indicates to the local CPU if the IPC11 event flag was set by the remote CPU. 0: No IPC11 event was set by the remote CPU 1: An IPC11 event was set by the remote CPU
10	IPC10	R	0h	Indicates to the local CPU if the IPC10 event flag was set by the remote CPU. 0: No IPC10 event was set by the remote CPU 1: An IPC10 event was set by the remote CPU
9	IPC9	R	0h	Indicates to the local CPU if the IPC9 event flag was set by the remote CPU. 0: No IPC9 event was set by the remote CPU 1: An IPC9 event was set by the remote CPU
8	IPC8	R	0h	Indicates to the local CPU if the IPC8 event flag was set by the remote CPU. 0: No IPC8 event was set by the remote CPU 1: An IPC8 event was set by the remote CPU
7	IPC7	R	0h	Indicates to the local CPU if the IPC7 event flag was set by the remote CPU. 0: No IPC7 event was set by the remote CPU 1: An IPC7 event was set by the remote CPU
6	IPC6	R	0h	Indicates to the local CPU if the IPC6 event flag was set by the remote CPU. 0: No IPC6 event was set by the remote CPU 1: An IPC6 event was set by the remote CPU
5	IPC5	R	0h	Indicates to the local CPU if the IPC5 event flag was set by the remote CPU. 0: No IPC5 event was set by the remote CPU 1: An IPC5 event was set by the remote CPU
4	IPC4	R	0h	Indicates to the local CPU if the IPC4 event flag was set by the remote CPU. 0: No IPC4 event was set by the remote CPU 1: An IPC4 event was set by the remote CPU
3	IPC3	R	0h	Indicates to the local CPU if the IPC3 event flag was set by the remote CPU. 0: No IPC3 event was set by the remote CPU 1: An IPC3 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R	0h	Indicates to the local CPU if the IPC2 event flag was set by the remote CPU. 0: No IPC2 event was set by the remote CPU 1: An IPC2 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R	0h	Indicates to the local CPU if the IPC1 event flag was set by the remote CPU. 0: No IPC1 event was set by the remote CPU 1: An IPC1 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

Table 6-6. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	IPC0	R	0h	<p>Indicates to the local CPU if the IPC0 event flag was set by the remote CPU.</p> <p>0: No IPC0 event was set by the remote CPU</p> <p>1: An IPC0 event was set by the remote CPU</p> <p>Notes</p> <p>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.</p>

6.8.2.3 IPCSET Register (Offset = 4h) [reset = 0h]

IPCSET is shown in [Figure 6-25](#) and described in [Table 6-26](#).

IPC remote flag set register

Figure 6-7. IPCSET Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-7. IPCSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit sets the IPC31 event flag for the remote CPU. Writing 0 has no effect.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit sets the IPC30 event flag for the remote CPU. Writing 0 has no effect.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit sets the IPC29 event flag for the remote CPU. Writing 0 has no effect.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit sets the IPC28 event flag for the remote CPU. Writing 0 has no effect.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit sets the IPC27 event flag for the remote CPU. Writing 0 has no effect.
26	IPC26	R=0/W=1	0h	Writing 1 to this bit sets the IPC26 event flag for the remote CPU. Writing 0 has no effect.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit sets the IPC25 event flag for the remote CPU. Writing 0 has no effect.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit sets the IPC24 event flag for the remote CPU. Writing 0 has no effect.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit sets the IPC23 event flag for the remote CPU. Writing 0 has no effect.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit sets the IPC22 event flag for the remote CPU. Writing 0 has no effect.
21	IPC21	R=0/W=1	0h	Writing 1 to this bit sets the IPC21 event flag for the remote CPU. Writing 0 has no effect.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit sets the IPC20 event flag for the remote CPU. Writing 0 has no effect.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit sets the IPC19 event flag for the remote CPU. Writing 0 has no effect.

Table 6-7. IPCSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18	IPC18	R=0/W=1	0h	Writing 1 to this bit sets the IPC18 event flag for the remote CPU. Writing 0 has no effect.
17	IPC17	R=0/W=1	0h	Writing 1 to this bit sets the IPC17 event flag for the remote CPU. Writing 0 has no effect.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit sets the IPC16 event flag for the remote CPU. Writing 0 has no effect.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit sets the IPC15 event flag for the remote CPU. Writing 0 has no effect.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit sets the IPC14 event flag for the remote CPU. Writing 0 has no effect.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit sets the IPC13 event flag for the remote CPU. Writing 0 has no effect.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit sets the IPC12 event flag for the remote CPU. Writing 0 has no effect.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit sets the IPC11 event flag for the remote CPU. Writing 0 has no effect.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit sets the IPC10 event flag for the remote CPU. Writing 0 has no effect.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit sets the IPC9 event flag for the remote CPU. Writing 0 has no effect.
8	IPC8	R=0/W=1	0h	Writing 1 to this bit sets the IPC8 event flag for the remote CPU. Writing 0 has no effect.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit sets the IPC7 event flag for the remote CPU. Writing 0 has no effect.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit sets the IPC6 event flag for the remote CPU. Writing 0 has no effect.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit sets the IPC5 event flag for the remote CPU. Writing 0 has no effect.
4	IPC4	R=0/W=1	0h	Writing 1 to this bit sets the IPC4 event flag for the remote CPU. Writing 0 has no effect.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit sets the IPC3 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit sets the IPC2 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit sets the IPC1 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

Table 6-7. IPCSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	IPC0	R=0/W=1	0h	<p>Writing 1 to this bit sets the IPC0 event flag for the remote CPU. Writing 0 has no effect.</p> <p>Notes:</p> <p>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.</p>

6.8.2.4 IPCCLR Register (Offset = 6h) [reset = 0h]

IPCCLR is shown in [Figure 6-26](#) and described in [Table 6-27](#).

IPC remote flag clear register

Figure 6-8. IPCCLR Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-8. IPCCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit clears the IPC31 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit clears the IPC30 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit clears the IPC29 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit clears the IPC28 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit clears the IPC27 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-8. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	IPC26	R=0/W=1	0h	Writing 1 to this bit clears the IPC26 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit clears the IPC25 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit clears the IPC24 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit clears the IPC23 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit clears the IPC22 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
21	IPC21	R=0/W=1	0h	Writing 1 to this bit clears the IPC21 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit clears the IPC20 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit clears the IPC19 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
18	IPC18	R=0/W=1	0h	Writing 1 to this bit clears the IPC18 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-8. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	IPC17	R=0/W=1	0h	Writing 1 to this bit clears the IPC17 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit clears the IPC16 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit clears the IPC15 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit clears the IPC14 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit clears the IPC13 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit clears the IPC12 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit clears the IPC11 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit clears the IPC10 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit clears the IPC9 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-8. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	IPC8	R=0/W=1	0h	Writing 1 to this bit clears the IPC8 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit clears the IPC7 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit clears the IPC6 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit clears the IPC5 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
4	IPC4	R=0/W=1	0h	Writing 1 to this bit clears the IPC4 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit clears the IPC3 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit clears the IPC2 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit clears the IPC1 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
0	IPC0	R=0/W=1	0h	Writing 1 to this bit clears the IPC0 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

6.8.2.5 IPCFLG Register (Offset = 8h) [reset = 0h]

IPCFLG is shown in [Figure 6-27](#) and described in [Table 6-28](#).

IPC remote flag status register

Figure 6-9. IPCFLG Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-9. IPCFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R	0h	Indicates to the local CPU whether the remote IPC31 event flag is set. 0: The remote IPC31 event flag is not set 1: The remote IPC31 event flag is set
30	IPC30	R	0h	Indicates to the local CPU whether the remote IPC30 event flag is set. 0: The remote IPC30 event flag is not set 1: The remote IPC30 event flag is set
29	IPC29	R	0h	Indicates to the local CPU whether the remote IPC29 event flag is set. 0: The remote IPC29 event flag is not set 1: The remote IPC29 event flag is set
28	IPC28	R	0h	Indicates to the local CPU whether the remote IPC28 event flag is set. 0: The remote IPC28 event flag is not set 1: The remote IPC28 event flag is set
27	IPC27	R	0h	Indicates to the local CPU whether the remote IPC27 event flag is set. 0: The remote IPC27 event flag is not set 1: The remote IPC27 event flag is set
26	IPC26	R	0h	Indicates to the local CPU whether the remote IPC26 event flag is set. 0: The remote IPC26 event flag is not set 1: The remote IPC26 event flag is set
25	IPC25	R	0h	Indicates to the local CPU whether the remote IPC25 event flag is set. 0: The remote IPC25 event flag is not set 1: The remote IPC25 event flag is set

Table 6-9. IPCFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	IPC24	R	0h	Indicates to the local CPU whether the remote IPC24 event flag is set. 0: The remote IPC24 event flag is not set 1: The remote IPC24 event flag is set
23	IPC23	R	0h	Indicates to the local CPU whether the remote IPC23 event flag is set. 0: The remote IPC23 event flag is not set 1: The remote IPC23 event flag is set
22	IPC22	R	0h	Indicates to the local CPU whether the remote IPC22 event flag is set. 0: The remote IPC22 event flag is not set 1: The remote IPC22 event flag is set
21	IPC21	R	0h	Indicates to the local CPU whether the remote IPC21 event flag is set. 0: The remote IPC21 event flag is not set 1: The remote IPC21 event flag is set
20	IPC20	R	0h	Indicates to the local CPU whether the remote IPC20 event flag is set. 0: The remote IPC20 event flag is not set 1: The remote IPC20 event flag is set
19	IPC19	R	0h	Indicates to the local CPU whether the remote IPC19 event flag is set. 0: The remote IPC19 event flag is not set 1: The remote IPC19 event flag is set
18	IPC18	R	0h	Indicates to the local CPU whether the remote IPC18 event flag is set. 0: The remote IPC18 event flag is not set 1: The remote IPC18 event flag is set
17	IPC17	R	0h	Indicates to the local CPU whether the remote IPC17 event flag is set. 0: The remote IPC17 event flag is not set 1: The remote IPC17 event flag is set
16	IPC16	R	0h	Indicates to the local CPU whether the remote IPC16 event flag is set. 0: The remote IPC16 event flag is not set 1: The remote IPC16 event flag is set
15	IPC15	R	0h	Indicates to the local CPU whether the remote IPC15 event flag is set. 0: The remote IPC15 event flag is not set 1: The remote IPC15 event flag is set
14	IPC14	R	0h	Indicates to the local CPU whether the remote IPC14 event flag is set. 0: The remote IPC14 event flag is not set 1: The remote IPC14 event flag is set
13	IPC13	R	0h	Indicates to the local CPU whether the remote IPC13 event flag is set. 0: The remote IPC13 event flag is not set 1: The remote IPC13 event flag is set
12	IPC12	R	0h	Indicates to the local CPU whether the remote IPC12 event flag is set. 0: The remote IPC12 event flag is not set 1: The remote IPC12 event flag is set

Table 6-9. IPCFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	IPC11	R	0h	Indicates to the local CPU whether the remote IPC11 event flag is set. 0: The remote IPC11 event flag is not set 1: The remote IPC11 event flag is set
10	IPC10	R	0h	Indicates to the local CPU whether the remote IPC10 event flag is set. 0: The remote IPC10 event flag is not set 1: The remote IPC10 event flag is set
9	IPC9	R	0h	Indicates to the local CPU whether the remote IPC9 event flag is set. 0: The remote IPC9 event flag is not set 1: The remote IPC9 event flag is set
8	IPC8	R	0h	Indicates to the local CPU whether the remote IPC8 event flag is set. 0: The remote IPC8 event flag is not set 1: The remote IPC8 event flag is set
7	IPC7	R	0h	Indicates to the local CPU whether the remote IPC7 event flag is set. 0: The remote IPC7 event flag is not set 1: The remote IPC7 event flag is set
6	IPC6	R	0h	Indicates to the local CPU whether the remote IPC6 event flag is set. 0: The remote IPC6 event flag is not set 1: The remote IPC6 event flag is set
5	IPC5	R	0h	Indicates to the local CPU whether the remote IPC5 event flag is set. 0: The remote IPC5 event flag is not set 1: The remote IPC5 event flag is set
4	IPC4	R	0h	Indicates to the local CPU whether the remote IPC4 event flag is set. 0: The remote IPC4 event flag is not set 1: The remote IPC4 event flag is set
3	IPC3	R	0h	Indicates to the local CPU whether the remote IPC3 event flag is set. 0: The remote IPC3 event flag is not set 1: The remote IPC3 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R	0h	Indicates to the local CPU whether the remote IPC2 event flag is set. 0: The remote IPC2 event flag is not set 1: The remote IPC2 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R	0h	Indicates to the local CPU whether the remote IPC1 event flag is set. 0: The remote IPC1 event flag is not set 1: The remote IPC1 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
0	IPC0	R	0h	Indicates to the local CPU whether the remote IPC0 event flag is set. 0: The remote IPC0 event flag is not set 1: The remote IPC0 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

6.8.2.6 IPCCOUNTERL Register (Offset = Ch) [reset = 0h]

IPCCOUNTERL is shown in [Figure 6-28](#) and described in [Table 6-29](#).

IPC Counter Low Register

Figure 6-10. IPCCOUNTERL Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-10. IPCCOUNTERL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R	0h	This is the lower 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.

6.8.2.7 IPCCOUNTERH Register (Offset = Eh) [reset = 0h]

IPCCOUNTERH is shown in [Figure 6-29](#) and described in [Table 6-30](#).

IPC Counter High Register

Figure 6-11. IPCCOUNTERH Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-11. IPCCOUNTERH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R	0h	This is the upper 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.

6.8.2.8 IPCSENDCOM Register (Offset = 10h) [reset = 0h]

IPCSENDCOM is shown in [Figure 6-34](#) and described in [Table 6-35](#).

Local to Remote IPC Command Register

Figure 6-12. IPCSENDCOM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-12. IPCSENDCOM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COMMAND	R/W	0h	<p>This is a general purpose register used to send software-defined commands to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDCOM is the same physical register as the remote CPU's IPCRECVCOM, and is located at the same address in both CPUs.</p>

6.8.2.9 IPCSENDADDR Register (Offset = 12h) [reset = 0h]

IPCSENDADDR is shown in [Figure 6-35](#) and described in [Table 6-36](#).

Local to Remote IPC Address Register

Figure 6-13. IPCSENDADDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-13. IPCSENDADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	0h	<p>This is a general purpose register used to send software-defined addresses to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDADDR is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.</p>

6.8.2.10 IPCSENDDATA Register (Offset = 14h) [reset = 0h]

IPCSENDDATA is shown in [Figure 6-36](#) and described in [Table 6-37](#).

Local to Remote IPC Data Register

Figure 6-14. IPCSENDDATA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-14. IPCSENDDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDATA	R/W	0h	<p>This is a general purpose register used to send software-defined data to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDDATA is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.</p>

6.8.2.11 IPCREMOTEREPLY Register (Offset = 16h) [reset = 0h]

IPCREMOTEREPLY is shown in [Figure 6-37](#) and described in [Table 6-38](#).

Remote to Local IPC Reply Data Register

Figure 6-15. IPCREMOTEREPLY Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-15. IPCREMOTEREPLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RDATA	R	0h	<p>This is a general purpose register used to receive software-defined data from the remote CPU's response to a command. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCREMOTEREPLY is the same physical register as the remote CPU's IPCLOCALREPLY, and is located at the same address in both CPUs.</p>

6.8.2.12 IPCRECVCOM Register (Offset = 18h) [reset = 0h]

IPCRECVCOM is shown in [Figure 6-30](#) and described in [Table 6-31](#).

Remote to Local IPC Command Register

Figure 6-16. IPCRECVCOM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-16. IPCRECVCOM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COMMAND	R	0h	<p>This is a general purpose register used to receive software-defined commands from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVCOM is the same physical register as the remote CPU's IPCSEND COM, and is located at the same address in both CPUs.</p>

6.8.2.13 IPCRECVADDR Register (Offset = 1Ah) [reset = 0h]

IPCRECVADDR is shown in [Figure 6-31](#) and described in [Table 6-32](#).

Remote to Local IPC Address Register

Figure 6-17. IPCRECVADDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-17. IPCRECVADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R	0h	<p>This is a general purpose register used to receive software-defined addresses from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVADDR is the same physical register as the remote CPU's IPCSENDADDR, and is located at the same address in both CPUs.</p>

6.8.2.14 IPCRECVDATA Register (Offset = 1Ch) [reset = 0h]

IPCRECVDATA is shown in [Figure 6-32](#) and described in [Table 6-33](#).

Remote to Local IPC Data Register

Figure 6-18. IPCRECVDATA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-18. IPCRECVDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDATA	R	0h	<p>This is a general purpose register used to receive software-defined data from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVDATA is the same physical register as the remote CPU's IPCSENDDATA, and is located at the same address in both CPUs.</p>

6.8.2.15 IPCLOCALREPLY Register (Offset = 1Eh) [reset = 0h]

IPCLOCALREPLY is shown in [Figure 6-33](#) and described in [Table 6-34](#).

Local to Remote IPC Reply Data Register

Figure 6-19. IPCLOCALREPLY Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-19. IPCLOCALREPLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RDATA	R/W	0h	<p>This is a general purpose register used to send software-defined data to the remote CPU in response to a command. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCLOCALREPLY is the same physical register as the remote CPU's IPCREMOТЕРEPLY, and is located at the same address in both CPUs.</p>

6.8.2.16 IPCBOOTSTS Register (Offset = 20h) [reset = 0h]

IPCBOOTSTS is shown in [Figure 6-38](#) and described in [Table 6-39](#).

CPU2 to CPU1 IPC Boot Status Register

Figure 6-20. IPCBOOTSTS Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-20. IPCBOOTSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BOOTSTS	R/W	0h	This register is used by CPU2 to pass the boot Status to CPU1. The data format is software-defined. It can only be written by CPU2.

6.8.2.17 IPCBOOTMODE Register (Offset = 22h) [reset = 0h]

IPCBOOTMODE is shown in [Figure 6-39](#) and described in [Table 6-40](#).

CPU1 to CPU2 IPC Boot Mode Register

Figure 6-21. IPCBOOTMODE Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-21. IPCBOOTMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BOOTMODE	R/W	0h	This register is used by CPU1 to pass a boot mode information to CPU2. The data format is software-defined. It can only be written by CPU1.

6.8.2.18 PUMPREQUEST Register (Offset = 24h) [reset = 0h]

PUMPREQUEST is shown in [Figure 6-40](#) and described in [Table 6-41](#).

Flash programming semaphore PUMP request register

Figure 6-22. PUMPREQUEST Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-22. PUMPREQUEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	R=0/W	0h	In order to write to the semaphore bits, 0x5a5a must be written to these key bits at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every semaphore change.
15-2	RESERVED	R=0	0h	Reserved
1-0	SEM	R/W	0h	These bits decide which CPU has control of the flash pump, which allows write access to the flash memory. The possible values are: 00 or 11: Read-only state. CPU1 has control of the pump, but CPU2 may seize control at any time. 01: CPU2 has exclusive control of the pump and of these semaphore bits. CPU2 can relinquish control by setting the bits back to 00 or 11. 10: CPU1 has exclusive control of the pump and of these semaphore bits. CPU1 can relinquish control by setting the bits back to 00 or 11. Going from 01->10 or 10->01 is not allowed. Going from 00->11 or 11->00 is allowed, but has no effect. The semaphore bits [1:0] must be written along with the correct key in bits [31:16].

6.8.3 IPC_REGS_CPU2 Registers

Table 6-23 lists the memory-mapped registers for the IPC_REGS_CPU2. All register offset addresses not listed in Table 6-23 should be considered as reserved locations and the register contents should not be modified.

Table 6-23. IPC_REGS_CPU2 Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	IPCACK	IPC incoming flag clear (acknowledge) register		Go
2h	IPCSTS	IPC incoming flag status register		Go
4h	IPCSET	IPC remote flag set register		Go
6h	IPCCLR	IPC remote flag clear register		Go
8h	IPCFLG	IPC remote flag status register		Go
Ch	IPCCOUNTERL	IPC Counter Low Register		Go
Eh	IPCCOUNTERH	IPC Counter High Register		Go
10h	IPCRECVCOM	Remote to Local IPC Command Register		Go
12h	IPCRECVADDR	Remote to Local IPC Address Register		Go
14h	IPCRECVDATA	Remote to Local IPC Data Register		Go
16h	IPCLOCALREPLY	Local to Remote IPC Reply Data Register		Go
18h	IPSENDCOM	Local to Remote IPC Command Register		Go
1Ah	IPSENDADDR	Local to Remote IPC Address Register		Go
1Ch	IPSENDATA	Local to Remote IPC Data Register		Go
1Eh	IPCREMOTEREPLY	Remote to Local IPC Reply Data Register		Go
20h	IPCBOOTSTS	CPU2 to CPU1 IPC Boot Status Register		Go
22h	IPCBOOTMODE	CPU1 to CPU2 IPC Boot Mode Register		Go
24h	PUMPREQUEST	Flash programming semaphore PUMP request register	EALLOW	Go

6.8.3.1 IPCACK Register (Offset = 0h) [reset = 0h]

IPCACK is shown in [Figure 6-23](#) and described in [Table 6-24](#).

IPC incoming flag clear (acknowledge) register

Figure 6-23. IPCACK Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-24. IPCACK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit clears the IPC31 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit clears the IPC30 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit clears the IPC29 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit clears the IPC28 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit clears the IPC27 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
26	IPC26	R=0/W=1	0h	Writing 1 to this bit clears the IPC26 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit clears the IPC25 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit clears the IPC24 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit clears the IPC23 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit clears the IPC22 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

Table 6-24. IPCACK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21	IPC21	R=0/W=1	0h	Writing 1 to this bit clears the IPC21 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit clears the IPC20 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit clears the IPC19 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
18	IPC18	R=0/W=1	0h	Writing 1 to this bit clears the IPC18 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
17	IPC17	R=0/W=1	0h	Writing 1 to this bit clears the IPC17 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit clears the IPC16 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit clears the IPC15 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit clears the IPC14 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit clears the IPC13 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit clears the IPC12 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit clears the IPC11 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit clears the IPC10 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit clears the IPC9 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
8	IPC8	R=0/W=1	0h	Writing 1 to this bit clears the IPC8 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit clears the IPC7 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit clears the IPC6 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit clears the IPC5 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

Table 6-24. IPCACK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	IPC4	R=0/W=1	0h	Writing 1 to this bit clears the IPC4 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit clears the IPC3 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit clears the IPC2 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit clears the IPC1 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.
0	IPC0	R=0/W=1	0h	Writing 1 to this bit clears the IPC0 event flag which was set by the remote CPU. Writing 0 to this bit has no effect.

6.8.3.2 IPCSTS Register (Offset = 2h) [reset = 0h]

IPCSTS is shown in [Figure 6-24](#) and described in [Table 6-25](#).

IPC incoming flag status register

Figure 6-24. IPCSTS Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-25. IPCSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R	0h	Indicates to the local CPU if the IPC31 event flag was set by the remote CPU. 0: No IPC31 event was set by the remote CPU 1: An IPC31 event was set by the remote CPU
30	IPC30	R	0h	Indicates to the local CPU if the IPC30 event flag was set by the remote CPU. 0: No IPC30 event was set by the remote CPU 1: An IPC30 event was set by the remote CPU
29	IPC29	R	0h	Indicates to the local CPU if the IPC29 event flag was set by the remote CPU. 0: No IPC29 event was set by the remote CPU 1: An IPC29 event was set by the remote CPU
28	IPC28	R	0h	Indicates to the local CPU if the IPC28 event flag was set by the remote CPU. 0: No IPC28 event was set by the remote CPU 1: An IPC28 event was set by the remote CPU
27	IPC27	R	0h	Indicates to the local CPU if the IPC27 event flag was set by the remote CPU. 0: No IPC27 event was set by the remote CPU 1: An IPC27 event was set by the remote CPU
26	IPC26	R	0h	Indicates to the local CPU if the IPC26 event flag was set by the remote CPU. 0: No IPC26 event was set by the remote CPU 1: An IPC26 event was set by the remote CPU
25	IPC25	R	0h	Indicates to the local CPU if the IPC25 event flag was set by the remote CPU. 0: No IPC25 event was set by the remote CPU 1: An IPC25 event was set by the remote CPU

Table 6-25. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	IPC24	R	0h	Indicates to the local CPU if the IPC24 event flag was set by the remote CPU. 0: No IPC24 event was set by the remote CPU 1: An IPC24 event was set by the remote CPU
23	IPC23	R	0h	Indicates to the local CPU if the IPC23 event flag was set by the remote CPU. 0: No IPC23 event was set by the remote CPU 1: An IPC23 event was set by the remote CPU
22	IPC22	R	0h	Indicates to the local CPU if the IPC22 event flag was set by the remote CPU. 0: No IPC22 event was set by the remote CPU 1: An IPC22 event was set by the remote CPU
21	IPC21	R	0h	Indicates to the local CPU if the IPC21 event flag was set by the remote CPU. 0: No IPC21 event was set by the remote CPU 1: An IPC21 event was set by the remote CPU
20	IPC20	R	0h	Indicates to the local CPU if the IPC20 event flag was set by the remote CPU. 0: No IPC20 event was set by the remote CPU 1: An IPC20 event was set by the remote CPU
19	IPC19	R	0h	Indicates to the local CPU if the IPC19 event flag was set by the remote CPU. 0: No IPC19 event was set by the remote CPU 1: An IPC19 event was set by the remote CPU
18	IPC18	R	0h	Indicates to the local CPU if the IPC18 event flag was set by the remote CPU. 0: No IPC18 event was set by the remote CPU 1: An IPC18 event was set by the remote CPU
17	IPC17	R	0h	Indicates to the local CPU if the IPC17 event flag was set by the remote CPU. 0: No IPC17 event was set by the remote CPU 1: An IPC17 event was set by the remote CPU
16	IPC16	R	0h	Indicates to the local CPU if the IPC16 event flag was set by the remote CPU. 0: No IPC16 event was set by the remote CPU 1: An IPC16 event was set by the remote CPU
15	IPC15	R	0h	Indicates to the local CPU if the IPC15 event flag was set by the remote CPU. 0: No IPC15 event was set by the remote CPU 1: An IPC15 event was set by the remote CPU
14	IPC14	R	0h	Indicates to the local CPU if the IPC14 event flag was set by the remote CPU. 0: No IPC14 event was set by the remote CPU 1: An IPC14 event was set by the remote CPU
13	IPC13	R	0h	Indicates to the local CPU if the IPC13 event flag was set by the remote CPU. 0: No IPC13 event was set by the remote CPU 1: An IPC13 event was set by the remote CPU
12	IPC12	R	0h	Indicates to the local CPU if the IPC12 event flag was set by the remote CPU. 0: No IPC12 event was set by the remote CPU 1: An IPC12 event was set by the remote CPU

Table 6-25. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	IPC11	R	0h	Indicates to the local CPU if the IPC11 event flag was set by the remote CPU. 0: No IPC11 event was set by the remote CPU 1: An IPC11 event was set by the remote CPU
10	IPC10	R	0h	Indicates to the local CPU if the IPC10 event flag was set by the remote CPU. 0: No IPC10 event was set by the remote CPU 1: An IPC10 event was set by the remote CPU
9	IPC9	R	0h	Indicates to the local CPU if the IPC9 event flag was set by the remote CPU. 0: No IPC9 event was set by the remote CPU 1: An IPC9 event was set by the remote CPU
8	IPC8	R	0h	Indicates to the local CPU if the IPC8 event flag was set by the remote CPU. 0: No IPC8 event was set by the remote CPU 1: An IPC8 event was set by the remote CPU
7	IPC7	R	0h	Indicates to the local CPU if the IPC7 event flag was set by the remote CPU. 0: No IPC7 event was set by the remote CPU 1: An IPC7 event was set by the remote CPU
6	IPC6	R	0h	Indicates to the local CPU if the IPC6 event flag was set by the remote CPU. 0: No IPC6 event was set by the remote CPU 1: An IPC6 event was set by the remote CPU
5	IPC5	R	0h	Indicates to the local CPU if the IPC5 event flag was set by the remote CPU. 0: No IPC5 event was set by the remote CPU 1: An IPC5 event was set by the remote CPU
4	IPC4	R	0h	Indicates to the local CPU if the IPC4 event flag was set by the remote CPU. 0: No IPC4 event was set by the remote CPU 1: An IPC4 event was set by the remote CPU
3	IPC3	R	0h	Indicates to the local CPU if the IPC3 event flag was set by the remote CPU. 0: No IPC3 event was set by the remote CPU 1: An IPC3 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R	0h	Indicates to the local CPU if the IPC2 event flag was set by the remote CPU. 0: No IPC2 event was set by the remote CPU 1: An IPC2 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R	0h	Indicates to the local CPU if the IPC1 event flag was set by the remote CPU. 0: No IPC1 event was set by the remote CPU 1: An IPC1 event was set by the remote CPU Notes [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

Table 6-25. IPCSTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	IPC0	R	0h	<p>Indicates to the local CPU if the IPC0 event flag was set by the remote CPU.</p> <p>0: No IPC0 event was set by the remote CPU</p> <p>1: An IPC0 event was set by the remote CPU</p> <p>Notes</p> <p>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.</p>

6.8.3.3 IPCSET Register (Offset = 4h) [reset = 0h]

IPCSET is shown in [Figure 6-25](#) and described in [Table 6-26](#).

IPC remote flag set register

Figure 6-25. IPCSET Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-26. IPCSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit sets the IPC31 event flag for the remote CPU. Writing 0 has no effect.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit sets the IPC30 event flag for the remote CPU. Writing 0 has no effect.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit sets the IPC29 event flag for the remote CPU. Writing 0 has no effect.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit sets the IPC28 event flag for the remote CPU. Writing 0 has no effect.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit sets the IPC27 event flag for the remote CPU. Writing 0 has no effect.
26	IPC26	R=0/W=1	0h	Writing 1 to this bit sets the IPC26 event flag for the remote CPU. Writing 0 has no effect.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit sets the IPC25 event flag for the remote CPU. Writing 0 has no effect.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit sets the IPC24 event flag for the remote CPU. Writing 0 has no effect.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit sets the IPC23 event flag for the remote CPU. Writing 0 has no effect.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit sets the IPC22 event flag for the remote CPU. Writing 0 has no effect.
21	IPC21	R=0/W=1	0h	Writing 1 to this bit sets the IPC21 event flag for the remote CPU. Writing 0 has no effect.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit sets the IPC20 event flag for the remote CPU. Writing 0 has no effect.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit sets the IPC19 event flag for the remote CPU. Writing 0 has no effect.

Table 6-26. IPCSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18	IPC18	R=0/W=1	0h	Writing 1 to this bit sets the IPC18 event flag for the remote CPU. Writing 0 has no effect.
17	IPC17	R=0/W=1	0h	Writing 1 to this bit sets the IPC17 event flag for the remote CPU. Writing 0 has no effect.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit sets the IPC16 event flag for the remote CPU. Writing 0 has no effect.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit sets the IPC15 event flag for the remote CPU. Writing 0 has no effect.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit sets the IPC14 event flag for the remote CPU. Writing 0 has no effect.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit sets the IPC13 event flag for the remote CPU. Writing 0 has no effect.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit sets the IPC12 event flag for the remote CPU. Writing 0 has no effect.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit sets the IPC11 event flag for the remote CPU. Writing 0 has no effect.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit sets the IPC10 event flag for the remote CPU. Writing 0 has no effect.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit sets the IPC9 event flag for the remote CPU. Writing 0 has no effect.
8	IPC8	R=0/W=1	0h	Writing 1 to this bit sets the IPC8 event flag for the remote CPU. Writing 0 has no effect.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit sets the IPC7 event flag for the remote CPU. Writing 0 has no effect.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit sets the IPC6 event flag for the remote CPU. Writing 0 has no effect.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit sets the IPC5 event flag for the remote CPU. Writing 0 has no effect.
4	IPC4	R=0/W=1	0h	Writing 1 to this bit sets the IPC4 event flag for the remote CPU. Writing 0 has no effect.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit sets the IPC3 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit sets the IPC2 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit sets the IPC1 event flag for the remote CPU. Writing 0 has no effect. Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

Table 6-26. IPCSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	IPC0	R=0/W=1	0h	<p>Writing 1 to this bit sets the IPC0 event flag for the remote CPU. Writing 0 has no effect.</p> <p>Notes:</p> <p>[1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.</p>

6.8.3.4 IPCCLR Register (Offset = 6h) [reset = 0h]

IPCCLR is shown in [Figure 6-26](#) and described in [Table 6-27](#).

IPC remote flag clear register

Figure 6-26. IPCCLR Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
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
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
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
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
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
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-27. IPCCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R=0/W=1	0h	Writing 1 to this bit clears the IPC31 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
30	IPC30	R=0/W=1	0h	Writing 1 to this bit clears the IPC30 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
29	IPC29	R=0/W=1	0h	Writing 1 to this bit clears the IPC29 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
28	IPC28	R=0/W=1	0h	Writing 1 to this bit clears the IPC28 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
27	IPC27	R=0/W=1	0h	Writing 1 to this bit clears the IPC27 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-27. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	IPC26	R=0/W=1	0h	Writing 1 to this bit clears the IPC26 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
25	IPC25	R=0/W=1	0h	Writing 1 to this bit clears the IPC25 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
24	IPC24	R=0/W=1	0h	Writing 1 to this bit clears the IPC24 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
23	IPC23	R=0/W=1	0h	Writing 1 to this bit clears the IPC23 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
22	IPC22	R=0/W=1	0h	Writing 1 to this bit clears the IPC22 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
21	IPC21	R=0/W=1	0h	Writing 1 to this bit clears the IPC21 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
20	IPC20	R=0/W=1	0h	Writing 1 to this bit clears the IPC20 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
19	IPC19	R=0/W=1	0h	Writing 1 to this bit clears the IPC19 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
18	IPC18	R=0/W=1	0h	Writing 1 to this bit clears the IPC18 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-27. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	IPC17	R=0/W=1	0h	Writing 1 to this bit clears the IPC17 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
16	IPC16	R=0/W=1	0h	Writing 1 to this bit clears the IPC16 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
15	IPC15	R=0/W=1	0h	Writing 1 to this bit clears the IPC15 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
14	IPC14	R=0/W=1	0h	Writing 1 to this bit clears the IPC14 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
13	IPC13	R=0/W=1	0h	Writing 1 to this bit clears the IPC13 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
12	IPC12	R=0/W=1	0h	Writing 1 to this bit clears the IPC12 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
11	IPC11	R=0/W=1	0h	Writing 1 to this bit clears the IPC11 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
10	IPC10	R=0/W=1	0h	Writing 1 to this bit clears the IPC10 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
9	IPC9	R=0/W=1	0h	Writing 1 to this bit clears the IPC9 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

Table 6-27. IPCCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	IPC8	R=0/W=1	0h	Writing 1 to this bit clears the IPC8 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
7	IPC7	R=0/W=1	0h	Writing 1 to this bit clears the IPC7 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
6	IPC6	R=0/W=1	0h	Writing 1 to this bit clears the IPC6 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
5	IPC5	R=0/W=1	0h	Writing 1 to this bit clears the IPC5 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
4	IPC4	R=0/W=1	0h	Writing 1 to this bit clears the IPC4 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
3	IPC3	R=0/W=1	0h	Writing 1 to this bit clears the IPC3 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
2	IPC2	R=0/W=1	0h	Writing 1 to this bit clears the IPC2 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
1	IPC1	R=0/W=1	0h	Writing 1 to this bit clears the IPC1 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.
0	IPC0	R=0/W=1	0h	Writing 1 to this bit clears the IPC0 flag for the remote CPU. Writing 0 has no effect. Notes: [1] Normally, each CPU will clear (acknowledge) only its own local flags. This mechanism may be useful if the remote CPU is non-responsive.

6.8.3.5 IPCFLG Register (Offset = 8h) [reset = 0h]

IPCFLG is shown in [Figure 6-27](#) and described in [Table 6-28](#).

IPC remote flag status register

Figure 6-27. IPCFLG Register

31	30	29	28	27	26	25	24
IPC31	IPC30	IPC29	IPC28	IPC27	IPC26	IPC25	IPC24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-28. IPCFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IPC31	R	0h	Indicates to the local CPU whether the remote IPC31 event flag is set. 0: The remote IPC31 event flag is not set 1: The remote IPC31 event flag is set
30	IPC30	R	0h	Indicates to the local CPU whether the remote IPC30 event flag is set. 0: The remote IPC30 event flag is not set 1: The remote IPC30 event flag is set
29	IPC29	R	0h	Indicates to the local CPU whether the remote IPC29 event flag is set. 0: The remote IPC29 event flag is not set 1: The remote IPC29 event flag is set
28	IPC28	R	0h	Indicates to the local CPU whether the remote IPC28 event flag is set. 0: The remote IPC28 event flag is not set 1: The remote IPC28 event flag is set
27	IPC27	R	0h	Indicates to the local CPU whether the remote IPC27 event flag is set. 0: The remote IPC27 event flag is not set 1: The remote IPC27 event flag is set
26	IPC26	R	0h	Indicates to the local CPU whether the remote IPC26 event flag is set. 0: The remote IPC26 event flag is not set 1: The remote IPC26 event flag is set
25	IPC25	R	0h	Indicates to the local CPU whether the remote IPC25 event flag is set. 0: The remote IPC25 event flag is not set 1: The remote IPC25 event flag is set

Table 6-28. IPCFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	IPC24	R	0h	Indicates to the local CPU whether the remote IPC24 event flag is set. 0: The remote IPC24 event flag is not set 1: The remote IPC24 event flag is set
23	IPC23	R	0h	Indicates to the local CPU whether the remote IPC23 event flag is set. 0: The remote IPC23 event flag is not set 1: The remote IPC23 event flag is set
22	IPC22	R	0h	Indicates to the local CPU whether the remote IPC22 event flag is set. 0: The remote IPC22 event flag is not set 1: The remote IPC22 event flag is set
21	IPC21	R	0h	Indicates to the local CPU whether the remote IPC21 event flag is set. 0: The remote IPC21 event flag is not set 1: The remote IPC21 event flag is set
20	IPC20	R	0h	Indicates to the local CPU whether the remote IPC20 event flag is set. 0: The remote IPC20 event flag is not set 1: The remote IPC20 event flag is set
19	IPC19	R	0h	Indicates to the local CPU whether the remote IPC19 event flag is set. 0: The remote IPC19 event flag is not set 1: The remote IPC19 event flag is set
18	IPC18	R	0h	Indicates to the local CPU whether the remote IPC18 event flag is set. 0: The remote IPC18 event flag is not set 1: The remote IPC18 event flag is set
17	IPC17	R	0h	Indicates to the local CPU whether the remote IPC17 event flag is set. 0: The remote IPC17 event flag is not set 1: The remote IPC17 event flag is set
16	IPC16	R	0h	Indicates to the local CPU whether the remote IPC16 event flag is set. 0: The remote IPC16 event flag is not set 1: The remote IPC16 event flag is set
15	IPC15	R	0h	Indicates to the local CPU whether the remote IPC15 event flag is set. 0: The remote IPC15 event flag is not set 1: The remote IPC15 event flag is set
14	IPC14	R	0h	Indicates to the local CPU whether the remote IPC14 event flag is set. 0: The remote IPC14 event flag is not set 1: The remote IPC14 event flag is set
13	IPC13	R	0h	Indicates to the local CPU whether the remote IPC13 event flag is set. 0: The remote IPC13 event flag is not set 1: The remote IPC13 event flag is set
12	IPC12	R	0h	Indicates to the local CPU whether the remote IPC12 event flag is set. 0: The remote IPC12 event flag is not set 1: The remote IPC12 event flag is set

Table 6-28. IPCFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	IPC11	R	0h	Indicates to the local CPU whether the remote IPC11 event flag is set. 0: The remote IPC11 event flag is not set 1: The remote IPC11 event flag is set
10	IPC10	R	0h	Indicates to the local CPU whether the remote IPC10 event flag is set. 0: The remote IPC10 event flag is not set 1: The remote IPC10 event flag is set
9	IPC9	R	0h	Indicates to the local CPU whether the remote IPC9 event flag is set. 0: The remote IPC9 event flag is not set 1: The remote IPC9 event flag is set
8	IPC8	R	0h	Indicates to the local CPU whether the remote IPC8 event flag is set. 0: The remote IPC8 event flag is not set 1: The remote IPC8 event flag is set
7	IPC7	R	0h	Indicates to the local CPU whether the remote IPC7 event flag is set. 0: The remote IPC7 event flag is not set 1: The remote IPC7 event flag is set
6	IPC6	R	0h	Indicates to the local CPU whether the remote IPC6 event flag is set. 0: The remote IPC6 event flag is not set 1: The remote IPC6 event flag is set
5	IPC5	R	0h	Indicates to the local CPU whether the remote IPC5 event flag is set. 0: The remote IPC5 event flag is not set 1: The remote IPC5 event flag is set
4	IPC4	R	0h	Indicates to the local CPU whether the remote IPC4 event flag is set. 0: The remote IPC4 event flag is not set 1: The remote IPC4 event flag is set
3	IPC3	R	0h	Indicates to the local CPU whether the remote IPC3 event flag is set. 0: The remote IPC3 event flag is not set 1: The remote IPC3 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
2	IPC2	R	0h	Indicates to the local CPU whether the remote IPC2 event flag is set. 0: The remote IPC2 event flag is not set 1: The remote IPC2 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
1	IPC1	R	0h	Indicates to the local CPU whether the remote IPC1 event flag is set. 0: The remote IPC1 event flag is not set 1: The remote IPC1 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.
0	IPC0	R	0h	Indicates to the local CPU whether the remote IPC0 event flag is set. 0: The remote IPC0 event flag is not set 1: The remote IPC0 event flag is set Notes: [1] IPC event flags 0-3 will trigger interrupts in the receiving CPU via the ePIE.

6.8.3.6 IPCCOUNTERL Register (Offset = Ch) [reset = 0h]

IPCCOUNTERL is shown in [Figure 6-28](#) and described in [Table 6-29](#).

IPC Counter Low Register

Figure 6-28. IPCCOUNTERL Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-29. IPCCOUNTERL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R	0h	This is the lower 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.

6.8.3.7 IPCCOUNTERH Register (Offset = Eh) [reset = 0h]

IPCCOUNTERH is shown in [Figure 6-29](#) and described in [Table 6-30](#).

IPC Counter High Register

Figure 6-29. IPCCOUNTERH Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-30. IPCCOUNTERH Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COUNT	R	0h	This is the upper 32-bits of free running 64 bit timestamp counter clocked by the PLLSYSCLK.

6.8.3.8 IPCRECVCOM Register (Offset = 10h) [reset = 0h]

IPCRECVCOM is shown in [Figure 6-30](#) and described in [Table 6-31](#).

Remote to Local IPC Command Register

Figure 6-30. IPCRECVCOM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-31. IPCRECVCOM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COMMAND	R	0h	<p>This is a general purpose register used to receive software-defined commands from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVCOM is the same physical register as the remote CPU's IPCSEND COM, and is located at the same address in both CPUs.</p>

6.8.3.9 IPCRECVADDR Register (Offset = 12h) [reset = 0h]

IPCRECVADDR is shown in [Figure 6-31](#) and described in [Table 6-32](#).

Remote to Local IPC Address Register

Figure 6-31. IPCRECVADDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-32. IPCRECVADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R	0h	<p>This is a general purpose register used to receive software-defined addresses from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVADDR is the same physical register as the remote CPU's IPCSENDADDR, and is located at the same address in both CPUs.</p>

6.8.3.10 IPCRECVDATA Register (Offset = 14h) [reset = 0h]

IPCRECVDATA is shown in [Figure 6-32](#) and described in [Table 6-33](#).

Remote to Local IPC Data Register

Figure 6-32. IPCRECVDATA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-33. IPCRECVDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDATA	R	0h	<p>This is a general purpose register used to receive software-defined data from the remote CPU. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCRECVDATA is the same physical register as the remote CPU's IPCSENDDATA, and is located at the same address in both CPUs.</p>

6.8.3.11 IPCLOCALREPLY Register (Offset = 16h) [reset = 0h]

IPCLOCALREPLY is shown in [Figure 6-33](#) and described in [Table 6-34](#).

Local to Remote IPC Reply Data Register

Figure 6-33. IPCLOCALREPLY Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-34. IPCLOCALREPLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RDATA	R/W	0h	<p>This is a general purpose register used to send software-defined data to the remote CPU in response to a command. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCLOCALREPLY is the same physical register as the remote CPU's IPCREMOREPLY, and is located at the same address in both CPUs.</p>

6.8.3.12 IPCSENDCOM Register (Offset = 18h) [reset = 0h]

IPCSENDCOM is shown in [Figure 6-34](#) and described in [Table 6-35](#).

Local to Remote IPC Command Register

Figure 6-34. IPCSENDCOM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-35. IPCSENDCOM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	COMMAND	R/W	0h	<p>This is a general purpose register used to send software-defined commands to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDCOM is the same physical register as the remote CPU's IPCRECVCOM, and is located at the same address in both CPUs.</p>

6.8.3.13 IPCSENDADDR Register (Offset = 1Ah) [reset = 0h]

IPCSENDADDR is shown in [Figure 6-35](#) and described in [Table 6-36](#).

Local to Remote IPC Address Register

Figure 6-35. IPCSENDADDR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-36. IPCSENDADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	0h	<p>This is a general purpose register used to send software-defined addresses to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDADDR is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.</p>

6.8.3.14 IPCSENDDATA Register (Offset = 1Ch) [reset = 0h]

IPCSSENDDATA is shown in [Figure 6-36](#) and described in [Table 6-37](#).

Local to Remote IPC Data Register

Figure 6-36. IPCSENDDATA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-37. IPCSENDDATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	WDATA	R/W	0h	<p>This is a general purpose register used to send software-defined data to the remote CPU. It can only be written by the local CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCSENDDATA is the same physical register as the remote CPU's IPCRECVDATA, and is located at the same address in both CPUs.</p>

6.8.3.15 IPCREMOTEREPLY Register (Offset = 1Eh) [reset = 0h]

IPCREMOTEREPLY is shown in [Figure 6-37](#) and described in [Table 6-38](#).

Remote to Local IPC Reply Data Register

Figure 6-37. IPCREMOTEREPLY Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-38. IPCREMOTEREPLY Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RDATA	R	0h	<p>This is a general purpose register used to receive software-defined data from the remote CPU's response to a command. It can only be written by the remote CPU.</p> <p>Notes</p> <p>[1] The local CPU's IPCREMOTEREPLY is the same physical register as the remote CPU's IPCLOCALREPLY, and is located at the same address in both CPUs.</p>

6.8.3.16 IPCBOOTSTS Register (Offset = 20h) [reset = 0h]

IPCBOOTSTS is shown in [Figure 6-38](#) and described in [Table 6-39](#).

CPU2 to CPU1 IPC Boot Status Register

Figure 6-38. IPCBOOTSTS Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-39. IPCBOOTSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BOOTSTS	R/W	0h	This register is used by CPU2 to pass the boot Status to CPU1. The data format is software-defined. It can only be written by CPU2.

6.8.3.17 IPCBOOTMODE Register (Offset = 22h) [reset = 0h]

IPCBOOTMODE is shown in [Figure 6-39](#) and described in [Table 6-40](#).

CPU1 to CPU2 IPC Boot Mode Register

Figure 6-39. IPCBOOTMODE Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-40. IPCBOOTMODE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BOOTMODE	R/W	0h	This register is used by CPU1 to pass a boot mode information to CPU2. The data format is software-defined. It can only be written by CPU1.

6.8.3.18 PUMPREQUEST Register (Offset = 24h) [reset = 0h]

PUMPREQUEST is shown in [Figure 6-40](#) and described in [Table 6-41](#).

Flash programming semaphore PUMP request register

Figure 6-40. PUMPREQUEST Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 6-41. PUMPREQUEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	R=0/W	0h	In order to write to the semaphore bits, 0x5a5a must be written to these key bits at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every semaphore change.
15-2	RESERVED	R=0	0h	Reserved
1-0	SEM	R/W	0h	These bits decide which CPU has control of the flash pump, which allows write access to the flash memory. The possible values are: 00 or 11: Read-only state. CPU1 has control of the pump, but CPU2 may seize control at any time. 01: CPU2 has exclusive control of the pump and of these semaphore bits. CPU2 can relinquish control by setting the bits back to 00 or 11. 10: CPU1 has exclusive control of the pump and of these semaphore bits. CPU1 can relinquish control by setting the bits back to 00 or 11. Going from 01->10 or 10->01 is not allowed. Going from 00->11 or 11->00 is allowed, but has no effect. The semaphore bits [1:0] must be written along with the correct key in bits [31:16].

General-Purpose Input/Output (GPIO)

The GPIO module controls the device's digital I/O MUX, 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. You can qualify the input signals to remove unwanted noise.

Topic	Page
7.1 GPIO Overview	844
7.2 Configuration Overview	845
7.3 Digital General-Purpose I/O Control	845
7.4 Input Qualification	847
7.5 USB Signals	850
7.6 SPI Signals	850
7.7 GPIO and Peripheral Muxing	851
7.8 Internal Pullup Configuration Requirements	852
7.9 Output X-BAR	854
7.10 Input X-BAR	856
7.11 Registers	858

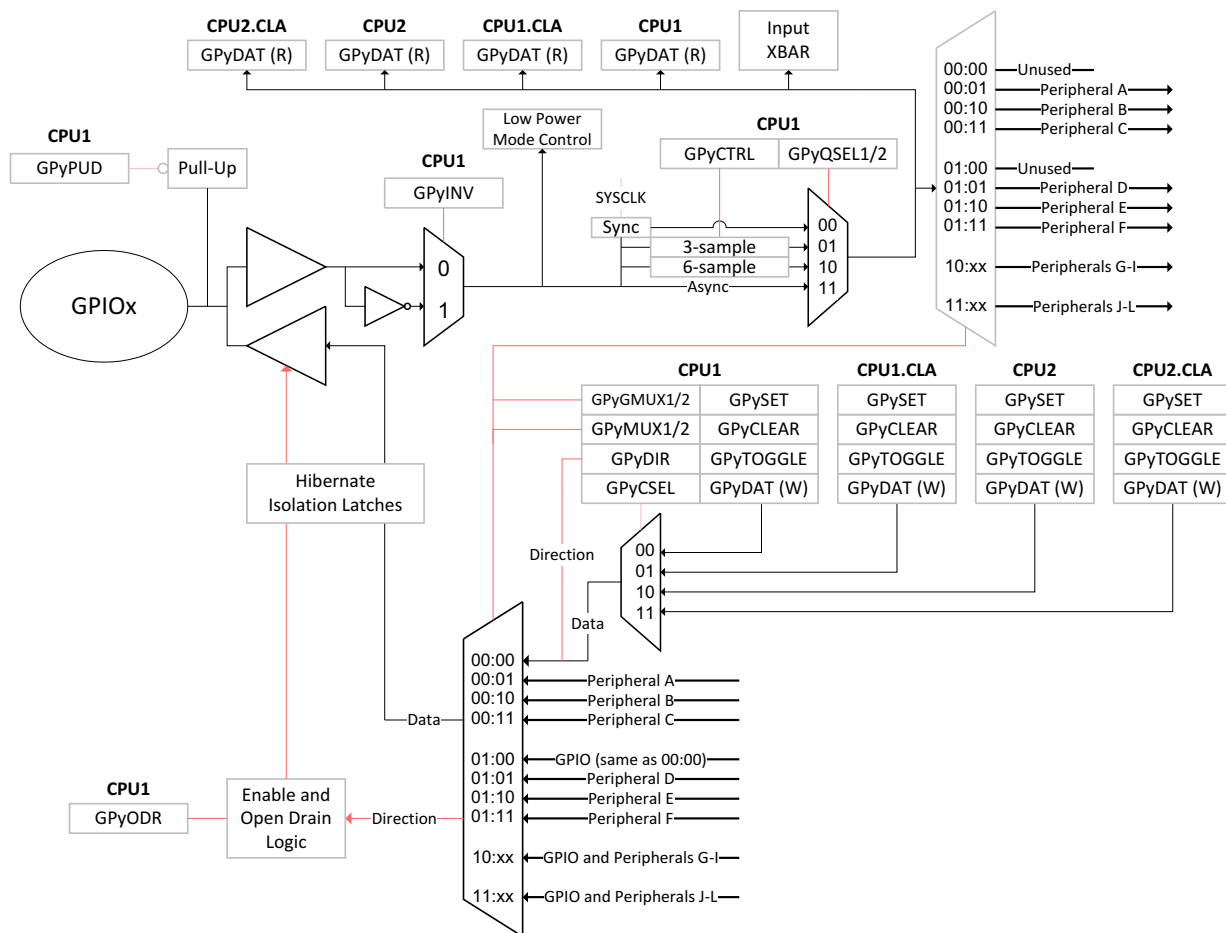
7.1 GPIO Overview

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 four CPU masters (CPU1, CPU1.CLA, CPU2, or CPU2.CLA). There are six I/O ports:

- Port A consists of GPIO0-GPIO31
- Port B consists of GPIO32-GPIO63
- Port C consists of GPIO64-GPIO95
- Port D consists of GPIO96-GPIO127
- Port E consists of GPIO128-GPIO159
- Port F consists of GPIO160-GPIO168

Figure 7-1 shows the GPIO logic for a single pin.

Figure 7-1. 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 [Section 7.6](#).

The USB PHY pin muxing is not shown in this diagram. For more details on USB pins, see [Section 7.5](#).

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 [Section 7.5](#) for details.

7.2 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. Note that GPIOs 42, 43, 46, and 47 are the only available USB pins. GPIO41 is hard-wired to be the wake-up signal in hibernate mode.

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 7.4](#).

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 0-63 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. In Hibernate mode, GPIO 41 is the only wake-up pin. 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 *ePWM* chapter of this manual.

7.3 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.

7.4 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.

7.4.1 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 I²C. 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 7.4.2](#).

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.

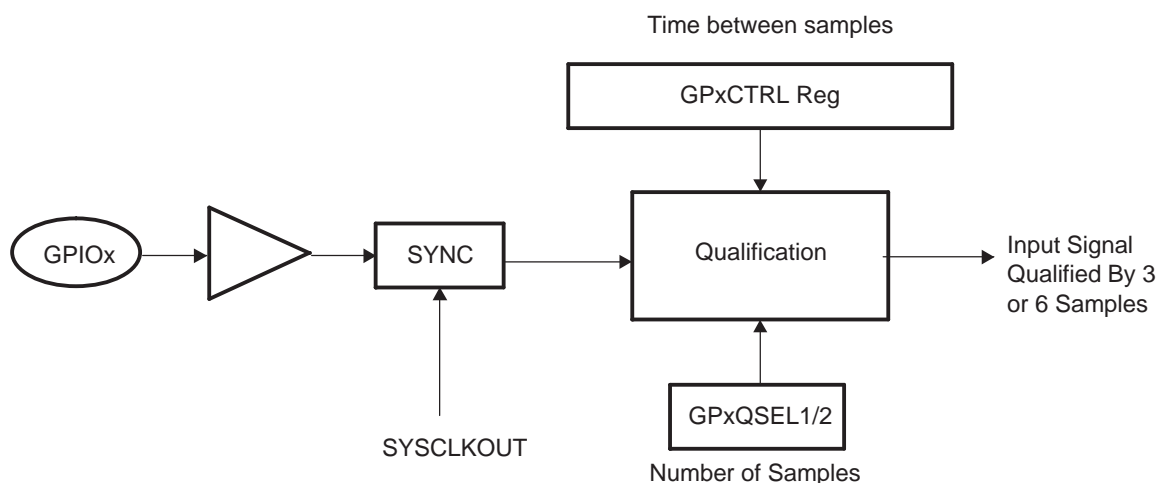
7.4.2 Synchronization to SYSCLKOUT Only

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.

7.4.3 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 7-2](#) and [Figure 7-3](#) 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 7-2. 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 GPACTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPACTRL[QUALPRD1]. [Table 7-1](#) and [Table 7-2](#) show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

Table 7-1. Sampling Period

	Sampling Period
If GPxCTRL[QUALPRDn] = 0	$1 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] \neq 0	$2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
	Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT

Table 7-2. Sampling Frequency

	Sampling Frequency
If GPxCTRL[QUALPRDn] = 0	$f_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] \neq 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 μs .

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 7-3](#). 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 7-1](#). Likewise, for a six-sample window, the sampling window width is five sampling periods wide. [Table 7-3](#) and [Table 7-4](#) 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 7-3. 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 7-4. 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.

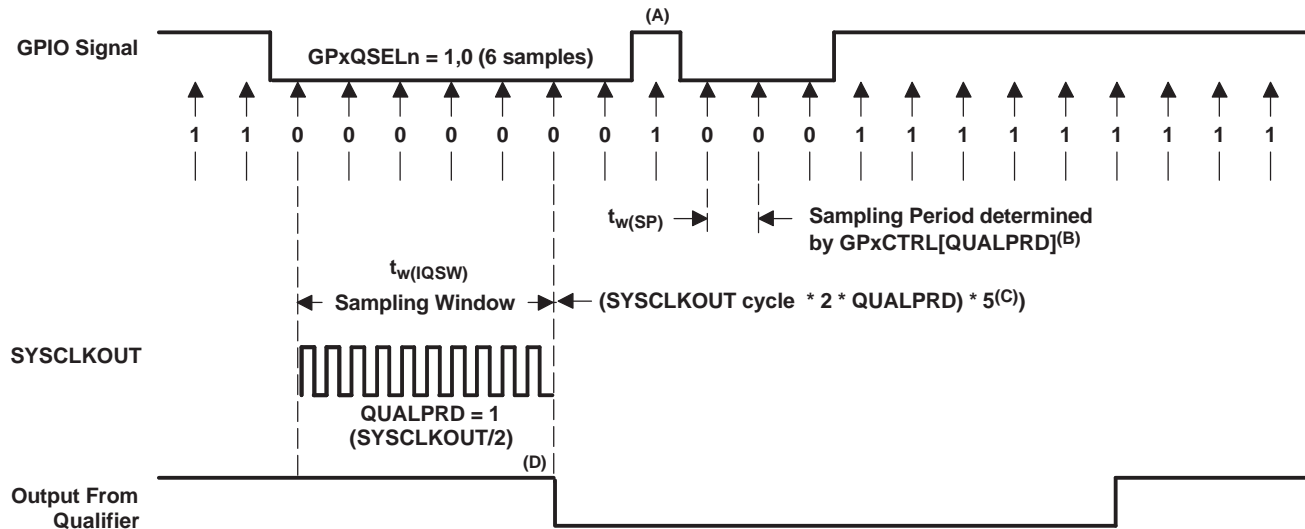
Example Qualification Window:

For the example shown in [Figure 7-3](#), 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 $t_w(\text{SP}) = 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$.

This configuration results in the following:

- The width of the sampling window is: .
 $t_w(\text{IQSW}) = 5 \times t_w(\text{SP}) = 5 \times 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$ or $5 \times 2 \times T_{\text{SYSCLKOUT}}$
- If, for example, $T_{\text{SYSCLKOUT}} = 16.67 \text{ ns}$, then the duration of the sampling window is:
 $t_w(\text{IQSW}) = 5 \times 2 \times 16.67 \text{ ns} = 166.7 \text{ ns}$.
- To account for the asynchronous nature of the input relative to the sampling period and SYSCLKOUT, up to an additional sampling period, $t_w(\text{SP})$, + $T_{\text{SYSCLKOUT}}$ may be required to detect a change in the input signal. For this example:
 $t_w(\text{SP}) + T_{\text{SYSCLKOUT}} = 333.4 \text{ ns} + 166.67 \text{ ns} = 500.1 \text{ ns}$
- In [Figure 7-3](#), the glitch (A) is shorter than the qualification window and will be ignored by the input qualifier.

Figure 7-3. 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 SYSCLKOUT cycle. For any other value "n", the qualification sampling period in 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT 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 SYSCLKOUT cycles or greater. In other words, the inputs should be stable for $(5 \times \text{QUALPRD} \times 2)$ SYSCLKOUT cycles. That would ensure 5 sampling periods for detection to occur. Since external signals are driven asynchronously, an 13-SYSCLKOUT-wide pulse ensures reliable recognition.

7.5 USB Signals

The USB module on this device has an internal physical layer transceiver (PHY). Its I/O signals are not normal digital signals, and as a result, they do not connect to the pins through the normal GPIO mux path. Instead, a special analog mux is used. To connect the USB signals to the device pins, set the GPyAMSEL bits appropriately as shown in [Table 7-5](#). Do not enable pullups or any other special pin option when using the USB signals.

Table 7-5. USB I/O Signal Muxing

Signal	GPIO	AMSEL
USBDM	42	GPBAMSEL[10]
USBDP	43	GPBAMSEL[11]

7.6 SPI Signals

The SPI module on this device has a high-speed mode that enables 40 Mbps communication. To achieve the highest possible speed, a special GPIO configuration is used on a single GPIO mux option for each SPI. These GPIOs may also be used by the SPI when not in high-speed mode (HS_MODE = 0). [Table 7-6](#) shows which GPIOs have the special mux option to allow SPI high-speed mode.

Table 7-6. High-Speed SPI-Enabled GPIOs

SPI pin	SPIA	SPIB	SPIC
SPISIMO	GPIO58	GPIO63	GPIO69
SPISOMI	GPIO59	GPIO64	GPIO70
SPICLK	GPIO60	GPIO65	GPIO71
SPISTE	GPIO61	GPIO66	GPIO72

To select these mux options the user should configure GPyGMUX and GPyMUX registers as shown in [Table 7-7](#).

Table 7-7. GPIO Configuration for High-Speed SPI

GPIO	SPI Signal	Mux Configuration	
GPIO58	SPISIMOA	GPBGMUX2[21:20]=11	GPBMUX2[21:20]=11
GPIO59	SPISOMIA	GPBGMUX2[23:22]=11	GPBMUX2[23:22]=11
GPIO60	SPICLKA	GPBGMUX2[25:24]=11	GPBMUX2[25:24]=11
GPIO61	SPISTEA	GPBGMUX2[27:26]=11	GPBMUX2[27:26]=11
GPIO63	SPISIMOB	GPBGMUX2[31:30]=11	GPBMUX2[31:30]=11
GPIO64	SPISOMIB	GPCGMUX1[1:0]=11	GPCMUX1[1:0]=11
GPIO65	SPICLKB	GPCGMUX1[3:2]=11	GPCMUX1[3:2]=11
GPIO66	SPISTEB	GPCGMUX1[5:4]=11	GPCMUX1[5:4]=11
GPIO69	SPISIMOC	GPCGMUX1[11:10]=11	GPCMUX1[11:10]=11
GPIO70	SPISOMIC	GPCGMUX1[13:12]=11	GPCMUX1[13:12]=11
GPIO71	SPICLKC	GPCGMUX1[15:14]=11	GPCMUX1[15:14]=11
GPIO72	SPISTEC	GPCGMUX1[17:16]=11	GPCMUX1[17:16]=11

7.7 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 device datasheet for muxing combinations and definitions.

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 7-8](#).

Table 7-8. GPIO and Peripheral Muxing

GPAGMUX1[13:12]	GPAMUX1[13:12]	Pin functionality
00	00	GPIO
00	01	EPWM4A
00	10	OUTPUTXBAR4
00	11	EXTSYNCOUT
01	00	GPIO
01	01	EQEP3A
01	10	CANTXB
10	00	GPIO
11	00	GPIO
All others		Reserved

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, XTRIPOUT1 can be assigned to GPIOs 2, 24, 34, or 58, depending on individual system requirements. An example of this is shown in [Table 7-9](#).

Table 7-9. 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 no pin is configured as an input to a peripheral, or if more than one pin is configured to an input for the same peripheral, then that input will be set to a hard-wired default value.

7.8 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 in the 176PTP or 100PZP 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.

For 176-pin packages, pullup resistors should be enabled on the following GPIOs:

Table 7-10. Pullup Resistors for 176-pin Packages

Unbonded GPIOs on the 176-pin package							
95	106	116	126	137	147	157	167
96	107	117	127	138	148	158	168
97	108	118	128	139	149	159	
98	109	119	129	140	150	160	
100	110	120	130	141	151	161	
101	111	121	131	142	152	162	
102	112	122	132	143	153	163	
103	113	123	134	144	154	164	
104	114	124	135	145	155	165	
105	115	125	136	146	156	166	

For 100-pin packages, pullup resistors should be enabled on the following GPIOs:

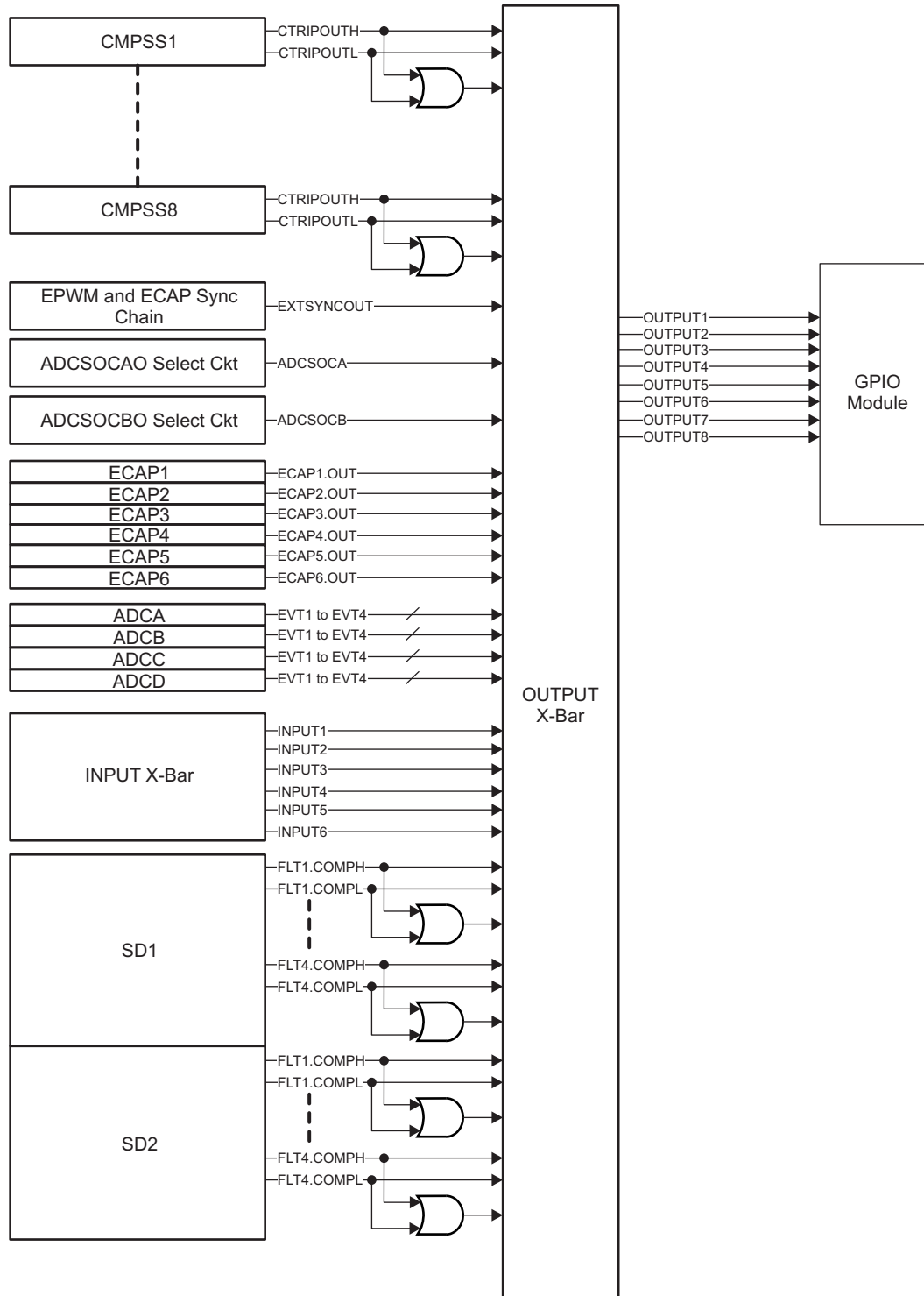
Table 7-11. Pullup Resistors for 100-pin Packages

Unbonded GPIOs on the 100-pin package								
0	30	48	77	101	116	131	146	161
1	31	49	79	102	117	132	147	162
5	32	50	80	103	118	133	148	163
6	33	51	81	104	119	134	149	164
7	34	52	82	105	120	135	150	165
8	35	53	83	106	121	136	151	166
9	36	54	88	107	122	137	152	167
22	37	55	93	108	123	138	153	168
23	38	56	94	109	124	139	154	
24	39	57	95	110	125	140	155	
25	40	67	96	111	126	141	156	
26	44	68	97	112	127	142	157	
27	45	74	98	113	128	143	158	
28	46	75	100	114	129	144	159	
29	47	76	101	115	130	145	160	

7.9 Output X-BAR

The GPIO module contains a X-BAR which allows an assortment of signals to be routed to a GPIO. The signals which are available to bring to the GPIO are listed in [Table 7-12](#). 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.

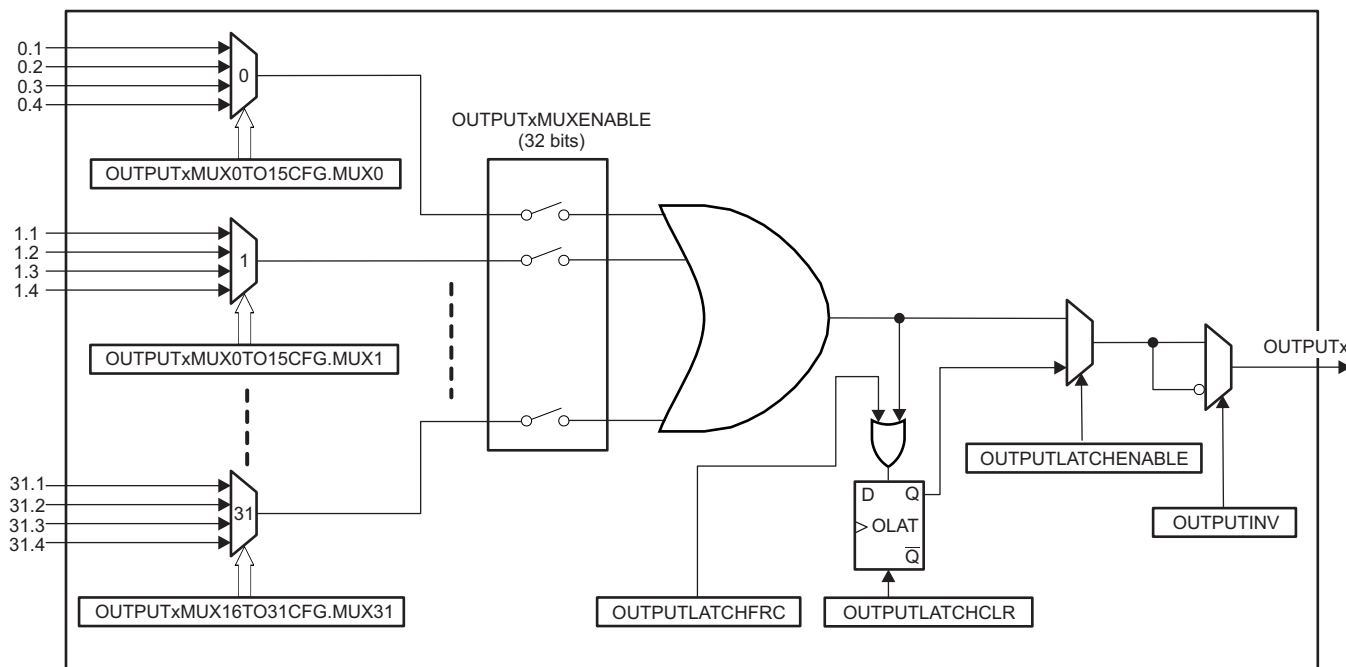
Figure 7-4. Output X- Bar



7.9.1 Output X-BAR Architecture

The Output X-BAR has eight outputs which are routed to the GPIO module. Figure 7-5 represents the architecture of a single output but it is identical to the architecture of all of the other outputs:

Figure 7-5. Output X-Bar Architecture



First, determine the signal(s) which should be passed to the GPIO by referencing Table 7-12. 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.

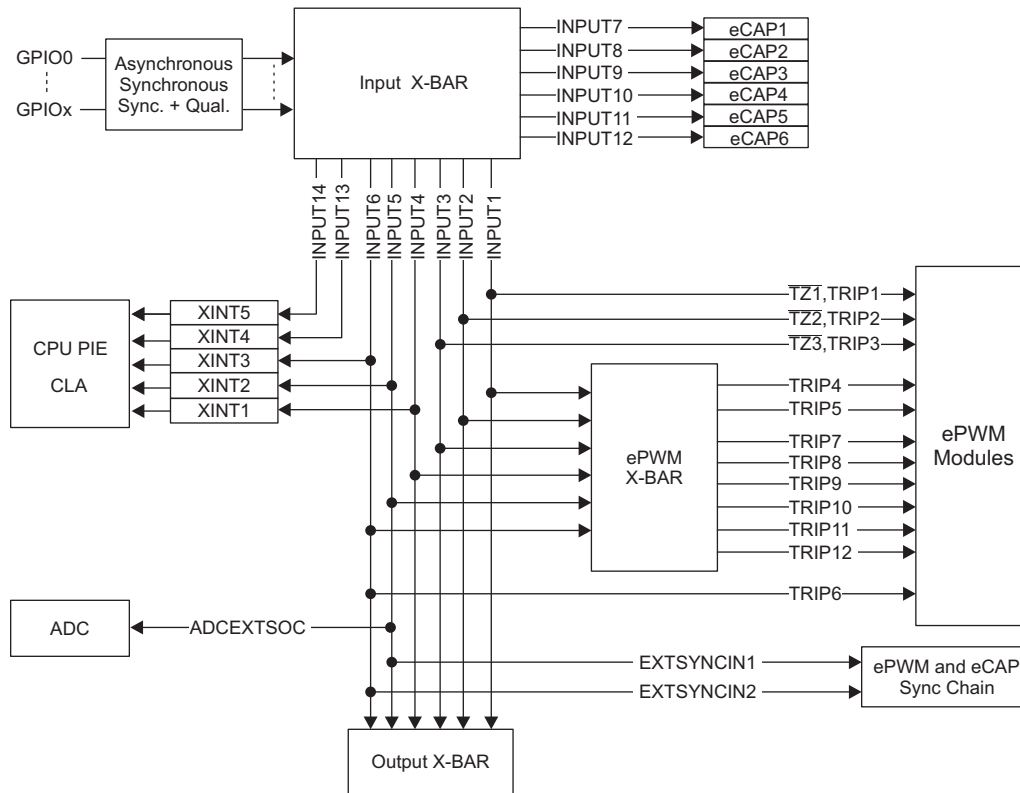
Table 7-12. Output X-Bar Mux Configuration Table

Mux	0	1	2	3
0	CMPSS1.CTRIPOUTH	CMPSS1.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT1	ECAP1.OUT
1	CMPSS1.CTRIPOUTL	INPUTXBAR1		ADCCEVT1
2	CMPSS2.CTRIPOUTH	CMPSS2.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT2	ECAP2.OUT
3	CMPSS2.CTRIPOUTL	INPUTXBAR2		ADCCEVT2
4	CMPSS3.CTRIPOUTH	CMPSS3.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT3	ECAP3.OUT
5	CMPSS3.CTRIPOUTL	INPUTXBAR3		ADCCEVT3
6	CMPSS4.CTRIPOUTH	CMPSS4.CTRIPOUTH_OR_CTRIPOUTL	ADCAEVT4	ECAP4.OUT
7	CMPSS4.CTRIPOUTL	INPUTXBAR4		ADCCEVT4
8	CMPSS5.CTRIPOUTH	CMPSS5.CTRIPOUTH_OR_CTRIPOUTL	ADCB EVT1	ECAP5.OUT
9	CMPSS5.CTRIPOUTL	INPUTXBAR5		ADCDEVT1
10	CMPSS6.CTRIPOUTH	CMPSS6.CTRIPOUTH_OR_CTRIPOUTL	ADCB EVT2	ECAP6.OUT
11	CMPSS6.CTRIPOUTL	INPUTXBAR6		ADCDEVT2
12	CMPSS7.CTRIPOUTH	CMPSS7.CTRIPOUTH_OR_CTRIPOUTL	ADCB EVT3	
13	CMPSS7.CTRIPOUTL	ADCSOCA		ADCDEVT3
14	CMPSS8.CTRIPOUTH	CMPSS8.CTRIPOUTH_OR_CTRIPOUTL	ADCB EVT4	EXTSYNCOUT
15	CMPSS8.CTRIPOUTL	ADCSOCB		ADCDEVT4
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL		
17	SD1FLT1.COMPL			
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL		
19	SD1FLT2.COMPL			
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL		
21	SD1FLT3.COMPL			
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL		
23	SD1FLT4.COMPL			
24	SD2FLT1.COMPH	SD2FLT1.COMPH_OR_COMPL		
25	SD2FLT1.COMPL			
26	SD2FLT2.COMPH	SD2FLT2.COMPH_OR_COMPL		
27	SD2FLT2.COMPL			
28	SD2FLT3.COMPH	SD2FLT3.COMPH_OR_COMPL		
29	SD2FLT3.COMPL			
30	SD2FLT4.COMPH	SD2FLT4.COMPH_OR_COMPL		
31	SD2FLT4.COMPL			

7.10 Input X-BAR

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. 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 7-6](#). For more information on configuration, see the TRIG_REGS register definitions at the end of this chapter.

Figure 7-6. Input X-BAR

Table 7-13. Input X-BAR Destinations

Input	Destinations
INPUT1	EPWM[TZ1,TRIP1], EPWM X-BAR, Output X-BAR
INPUT2	EPWM[TZ2,TRIP2], EPWM X-BAR, Output X-BAR
INPUT3	EPWM[TZ3,TRIP3], EPWM X-BAR, Output X-BAR
INPUT4	XINT1, EPWM X-BAR, Output X-BAR
INPUT5	XINT2, ADCEXTSOC, EXTSYNINCIN1, EPWM X-BAR, Output X-BAR
INPUT6	XINT3, EPWM[TRIP6], EXTSYNINCIN2, EPWM X-BAR, Output X-BAR
INPUT7	ECAP1
INPUT8	ECAP2
INPUT9	ECAP3
INPUT10	ECAP4
INPUT11	ECAP5
INPUT12	ECAP6
INPUT13	XINT4
INPUT14	XINT5

7.11 Registers

7.11.1 GPIO Base Addresses

Table 7-14. GPIO Base Address Table

Device Registers	Register Name	Start Address	End Address
InputXbarRegs ⁽¹⁾	INPUT_XBAR_REGS	0x0000_7900	0x0000_791F
XbarRegs ⁽¹⁾	XBAR_REGS	0x0000_7920	0x0000_793F
OutputXbarRegs ⁽¹⁾	OUTPUT_XBAR_REGS	0x0000_7A80	0x0000_7ABF
GpioCtrlRegs ⁽¹⁾	GPIO_CTRL_REGS	0x0000_7C00	0x0000_7D7F
GpioDataRegs	GPIO_DATA_REGS	0x0000_7F00	0x0000_7F2F

⁽¹⁾ Only available on CPU1.

7.1.1.2 INPUT_XBAR_REGS Registers

Table 7-15 lists the memory-mapped registers for the INPUT_XBAR_REGS. All register offset addresses not listed in Table 7-15 should be considered as reserved locations and the register contents should not be modified.

Table 7-15. INPUT_XBAR_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	INPUT1SELECT	INPUT1 Input Select Register (GPIO0 to x)	EALLOW	Go
1h	INPUT2SELECT	INPUT2 Input Select Register (GPIO0 to x)	EALLOW	Go
2h	INPUT3SELECT	INPUT3 Input Select Register (GPIO0 to x)	EALLOW	Go
3h	INPUT4SELECT	INPUT4 Input Select Register (GPIO0 to x)	EALLOW	Go
4h	INPUT5SELECT	INPUT5 Input Select Register (GPIO0 to x)	EALLOW	Go
5h	INPUT6SELECT	INPUT6 Input Select Register (GPIO0 to x)	EALLOW	Go
6h	INPUT7SELECT	INPUT7 Input Select Register (GPIO0 to x)	EALLOW	Go
7h	INPUT8SELECT	INPUT8 Input Select Register (GPIO0 to x)	EALLOW	Go
8h	INPUT9SELECT	INPUT9 Input Select Register (GPIO0 to x)	EALLOW	Go
9h	INPUT10SELECT	INPUT10 Input Select Register (GPIO0 to x)	EALLOW	Go
Ah	INPUT11SELECT	INPUT11 Input Select Register (GPIO0 to x)	EALLOW	Go
Bh	INPUT12SELECT	INPUT12 Input Select Register (GPIO0 to x)	EALLOW	Go
Ch	INPUT13SELECT	INPUT13 Input Select Register (GPIO0 to x)	EALLOW	Go
Dh	INPUT14SELECT	INPUT14 Input Select Register (GPIO0 to x)	EALLOW	Go
1Eh	INPUTSELECTLOCK	Input Select Lock Register	EALLOW	Go

7.11.2.1 INPUT1SELECT Register (Offset = 0h) [reset = 0h]

INPUT1SELECT is shown in [Figure 7-7](#) and described in [Table 7-16](#).

INPUT1 Input Select Register (GPIO0 to x)

Figure 7-7. INPUT1SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-16. INPUT1SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT1 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.2 INPUT2SELECT Register (Offset = 1h) [reset = 0h]

INPUT2SELECT is shown in [Figure 7-8](#) and described in [Table 7-17](#).

INPUT2 Input Select Register (GPIO0 to x)

Figure 7-8. INPUT2SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-17. INPUT2SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT2 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.3 INPUT3SELECT Register (Offset = 2h) [reset = 0h]

INPUT3SELECT is shown in [Figure 7-9](#) and described in [Table 7-18](#).

INPUT3 Input Select Register (GPIO0 to x)

Figure 7-9. INPUT3SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-18. INPUT3SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT3 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.4 INPUT4SELECT Register (Offset = 3h) [reset = 0h]

INPUT4SELECT is shown in [Figure 7-10](#) and described in [Table 7-19](#).

INPUT4 Input Select Register (GPIO0 to x)

Figure 7-10. INPUT4SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-19. INPUT4SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT4 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.5 INPUT5SELECT Register (Offset = 4h) [reset = 0h]

INPUT5SELECT is shown in [Figure 7-11](#) and described in [Table 7-20](#).

INPUT5 Input Select Register (GPIO0 to x)

Figure 7-11. INPUT5SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-20. INPUT5SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT5 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.6 INPUT6SELECT Register (Offset = 5h) [reset = 0h]

INPUT6SELECT is shown in [Figure 7-12](#) and described in [Table 7-21](#).

INPUT6 Input Select Register (GPIO0 to x)

Figure 7-12. INPUT6SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-21. INPUT6SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT6 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.7 INPUT7SELECT Register (Offset = 6h) [reset = 0h]

INPUT7SELECT is shown in [Figure 7-13](#) and described in [Table 7-22](#).

INPUT7 Input Select Register (GPIO0 to x)

Figure 7-13. INPUT7SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-22. INPUT7SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT7 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.8 INPUT8SELECT Register (Offset = 7h) [reset = 0h]

INPUT8SELECT is shown in [Figure 7-14](#) and described in [Table 7-23](#).

INPUT8 Input Select Register (GPIO0 to x)

Figure 7-14. INPUT8SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-23. INPUT8SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT8 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.9 INPUT9SELECT Register (Offset = 8h) [reset = 0h]

INPUT9SELECT is shown in [Figure 7-15](#) and described in [Table 7-24](#).

INPUT9 Input Select Register (GPIO0 to x)

Figure 7-15. INPUT9SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-24. INPUT9SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT9 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.10 INPUT10SELECT Register (Offset = 9h) [reset = 0h]

INPUT10SELECT is shown in [Figure 7-16](#) and described in [Table 7-25](#).

INPUT10 Input Select Register (GPIO0 to x)

Figure 7-16. INPUT10SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-25. INPUT10SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT10 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.11 INPUT11SELECT Register (Offset = Ah) [reset = 0h]

INPUT11SELECT is shown in [Figure 7-17](#) and described in [Table 7-26](#).

INPUT11 Input Select Register (GPIO0 to x)

Figure 7-17. INPUT11SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-26. INPUT11SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT11 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.12 INPUT12SELECT Register (Offset = Bh) [reset = 0h]

INPUT12SELECT is shown in [Figure 7-18](#) and described in [Table 7-27](#).

INPUT12 Input Select Register (GPIO0 to x)

Figure 7-18. INPUT12SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-27. INPUT12SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT12 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.13 INPUT13SELECT Register (Offset = Ch) [reset = 0h]

INPUT13SELECT is shown in [Figure 7-19](#) and described in [Table 7-28](#).

INPUT13 Input Select Register (GPIO0 to x)

Figure 7-19. INPUT13SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-28. INPUT13SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT13 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.14 INPUT14SELECT Register (Offset = Dh) [reset = 0h]

INPUT14SELECT is shown in [Figure 7-20](#) and described in [Table 7-29](#).

INPUT14 Input Select Register (GPIO0 to x)

Figure 7-20. INPUT14SELECT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SELECT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-29. INPUT14SELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	0h	Select GPIO for INPUT14 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xn : Select GPIO _n

7.11.2.15 INPUTSELECTLOCK Register (Offset = 1Eh) [reset = 0h]

INPUTSELECTLOCK is shown in [Figure 7-21](#) and described in [Table 7-30](#).

Input Select Lock Register

Figure 7-21. 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/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-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/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-30. INPUTSELECTLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15	INPUT16SELECT	R/SOnce	0h	Lock bit for INPUT16SELECT Register: 0: Respective register is not locked 1: Respective register is locked. 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 Notes: [1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.
14	INPUT15SELECT	R/SOnce	0h	Lock bit for INPUT15SELECT Register: 0: Respective register is not locked 1: Respective register is locked. 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 Notes: [1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.
13	INPUT14SELECT	R/SOnce	0h	Lock bit for INPUT14SELECT Register: 0: Respective register is not locked 1: Respective register is locked. 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 Notes: [1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.

Table 7-30. INPUTSELECTLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	INPUT13SELECT	R/SONce	0h	<p>Lock bit for INPUT13SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
11	INPUT12SELECT	R/SONce	0h	<p>Lock bit for INPUT12SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
10	INPUT11SELECT	R/SONce	0h	<p>Lock bit for INPUT11SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
9	INPUT10SELECT	R/SONce	0h	<p>Lock bit for INPUT11SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
8	INPUT9SELECT	R/SONce	0h	<p>Lock bit for INPUT10SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
7	INPUT8SELECT	R/SONce	0h	<p>Lock bit for INPUT9SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>

Table 7-30. INPUTSELECTLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	INPUT7SELECT	R/SONce	0h	<p>Lock bit for INPUT8SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
5	INPUT6SELECT	R/SONce	0h	<p>Lock bit for INPUT7SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
4	INPUT5SELECT	R/SONce	0h	<p>Lock bit for INPUT5SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
3	INPUT4SELECT	R/SONce	0h	<p>Lock bit for INPUT4SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
2	INPUT3SELECT	R/SONce	0h	<p>Lock bit for INPUT3SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>
1	INPUT2SELECT	R/SONce	0h	<p>Lock bit for INPUT2SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>

Table 7-30. INPUTSELECTLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INPUT1SELECT	R/SOnce	0h	<p>Lock bit for INPUT1SELECT Register:</p> <p>0: Respective register is not locked</p> <p>1: Respective register is locked.</p> <p>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</p> <p>Notes:</p> <p>[1] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed.</p>

7.11.3 OUTPUT_XBAR_REGS Registers

Table 7-31 lists the memory-mapped registers for the OUTPUT_XBAR_REGS. All register offset addresses not listed in Table 7-31 should be considered as reserved locations and the register contents should not be modified.

Table 7-31. OUTPUT_XBAR_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	OUTPUT1MUX0TO15CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	Go
2h	OUTPUT1MUX16TO31CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	Go
4h	OUTPUT2MUX0TO15CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	Go
6h	OUTPUT2MUX16TO31CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	Go
8h	OUTPUT3MUX0TO15CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	Go
Ah	OUTPUT3MUX16TO31CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	Go
Ch	OUTPUT4MUX0TO15CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	Go
Eh	OUTPUT4MUX16TO31CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	Go
10h	OUTPUT5MUX0TO15CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	Go
12h	OUTPUT5MUX16TO31CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	Go
14h	OUTPUT6MUX0TO15CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	Go
16h	OUTPUT6MUX16TO31CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	Go
18h	OUTPUT7MUX0TO15CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	Go
1Ah	OUTPUT7MUX16TO31CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	Go
1Ch	OUTPUT8MUX0TO15CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	Go
1Eh	OUTPUT8MUX16TO31CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	Go
20h	OUTPUT1MUXENABLE	Output X-BAR Mux Enable for Output 1	EALLOW	Go
22h	OUTPUT2MUXENABLE	Output X-BAR Mux Enable for Output 2	EALLOW	Go
24h	OUTPUT3MUXENABLE	Output X-BAR Mux Enable for Output 3	EALLOW	Go
26h	OUTPUT4MUXENABLE	Output X-BAR Mux Enable for Output 4	EALLOW	Go
28h	OUTPUT5MUXENABLE	Output X-BAR Mux Enable for Output 5	EALLOW	Go
2Ah	OUTPUT6MUXENABLE	Output X-BAR Mux Enable for Output 6	EALLOW	Go
2Ch	OUTPUT7MUXENABLE	Output X-BAR Mux Enable for Output 7	EALLOW	Go
2Eh	OUTPUT8MUXENABLE	Output X-BAR Mux Enable for Output 8	EALLOW	Go
30h	OUTPUTLATCH	Output X-BAR Output Latch		Go
32h	OUTPUTLATCHCLR	Output X-BAR Output Latch Clear		Go
34h	OUTPUTLATCHFRC	Output X-BAR Output Latch Clear		Go
36h	OUTPUTLATCHENABLE	Output X-BAR Output Latch Enable	EALLOW	Go
38h	OUTPUTINV	Output X-BAR Output Inversion	EALLOW	Go
3Eh	OUTPUTLOCK	Output X-BAR Configuration Lock register	EALLOW	Go

7.11.3.1 OUTPUT1MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

OUTPUT1MUX0TO15CFG is shown in [Figure 7-22](#) and described in [Table 7-32](#).

Output X-BAR Mux Configuration for Output 1

Figure 7-22. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-32. OUTPUT1MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

Table 7-32. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

Table 7-32. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

7.11.3.2 OUTPUT1MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

OUTPUT1MUX16TO31CFG is shown in [Figure 7-23](#) and described in [Table 7-33](#).

Output X-BAR Mux Configuration for Output 1

Figure 7-23. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-33. OUTPUT1MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT1 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

Table 7-33. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

Table 7-33. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT1 of OUTPUT-XBAR 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.

7.11.3.3 OUTPUT2MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

OUTPUT2MUX0TO15CFG is shown in [Figure 7-24](#) and described in [Table 7-34](#).

Output X-BAR Mux Configuration for Output 2

Figure 7-24. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-34. OUTPUT2MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

Table 7-34. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

Table 7-34. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

7.11.3.4 OUTPUT2MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

OUTPUT2MUX16TO31CFG is shown in [Figure 7-25](#) and described in [Table 7-35](#).

Output X-BAR Mux Configuration for Output 2

Figure 7-25. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-35. OUTPUT2MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT2 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

Table 7-35. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

Table 7-35. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT2 of OUTPUT-XBAR 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.

7.11.3.5 OUTPUT3MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

OUTPUT3MUX0TO15CFG is shown in [Figure 7-26](#) and described in [Table 7-36](#).

Output X-BAR Mux Configuration for Output 3

Figure 7-26. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-36. OUTPUT3MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

Table 7-36. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

Table 7-36. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

7.11.3.6 OUTPUT3MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

OUTPUT3MUX16TO31CFG is shown in [Figure 7-27](#) and described in [Table 7-37](#).

Output X-BAR Mux Configuration for Output 3

Figure 7-27. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-37. OUTPUT3MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT3 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

Table 7-37. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

Table 7-37. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT3 of OUTPUT-XBAR 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.

7.11.3.7 OUTPUT4MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

OUTPUT4MUX0TO15CFG is shown in [Figure 7-28](#) and described in [Table 7-38](#).

Output X-BAR Mux Configuration for Output 4

Figure 7-28. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-38. OUTPUT4MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

Table 7-38. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

Table 7-38. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

7.11.3.8 OUTPUT4MUX16TO31CFG Register (Offset = Eh) [reset = 0h]

OUTPUT4MUX16TO31CFG is shown in [Figure 7-29](#) and described in [Table 7-39](#).

Output X-BAR Mux Configuration for Output 4

Figure 7-29. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-39. OUTPUT4MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT4 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

Table 7-39. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

Table 7-39. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT4 of OUTPUT-XBAR 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.

7.11.3.9 OUTPUT5MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

OUTPUT5MUX0TO15CFG is shown in [Figure 7-30](#) and described in [Table 7-40](#).

Output X-BAR Mux Configuration for Output 5

Figure 7-30. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-40. OUTPUT5MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

Table 7-40. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

Table 7-40. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

7.11.3.10 OUTPUT5MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

OUTPUT5MUX16TO31CFG is shown in [Figure 7-31](#) and described in [Table 7-41](#).

Output X-BAR Mux Configuration for Output 5

Figure 7-31. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-41. OUTPUT5MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT5 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

Table 7-41. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

Table 7-41. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT5 of OUTPUT-XBAR 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.

7.11.3.11 OUTPUT6MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

OUTPUT6MUX0TO15CFG is shown in [Figure 7-32](#) and described in [Table 7-42](#).

Output X-BAR Mux Configuration for Output 6

Figure 7-32. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-42. OUTPUT6MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

Table 7-42. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

Table 7-42. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

7.11.3.12 OUTPUT6MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

OUTPUT6MUX16TO31CFG is shown in [Figure 7-33](#) and described in [Table 7-43](#).

Output X-BAR Mux Configuration for Output 6

Figure 7-33. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-43. OUTPUT6MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT6 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

Table 7-43. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

Table 7-43. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT6 of OUTPUT-XBAR 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.

7.11.3.13 OUTPUT7MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

OUTPUT7MUX0TO15CFG is shown in [Figure 7-34](#) and described in [Table 7-44](#).

Output X-BAR Mux Configuration for Output 7

Figure 7-34. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-44. OUTPUT7MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

Table 7-44. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

Table 7-44. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

7.11.3.14 OUTPUT7MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

OUTPUT7MUX16TO31CFG is shown in [Figure 7-35](#) and described in [Table 7-45](#).

Output X-BAR Mux Configuration for Output 7

Figure 7-35. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-45. OUTPUT7MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT7 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

Table 7-45. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

Table 7-45. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT7 of OUTPUT-XBAR 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.

7.11.3.15 OUTPUT8MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

OUTPUT8MUX0TO15CFG is shown in [Figure 7-36](#) and described in [Table 7-46](#).

Output X-BAR Mux Configuration for Output 8

Figure 7-36. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-46. OUTPUT8MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

Table 7-46. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

Table 7-46. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

7.11.3.16 OUTPUT8MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

OUTPUT8MUX16TO31CFG is shown in [Figure 7-37](#) and described in [Table 7-47](#).

Output X-BAR Mux Configuration for Output 8

Figure 7-37. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-47. OUTPUT8MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for OUTPUT8 of OUTPUT-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 Output X-BAR section of this chapter for more details.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

Table 7-47. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

Table 7-47. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for OUTPUT8 of OUTPUT-XBAR 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.

7.11.3.17 OUTPUT1MUXENABLE Register (Offset = 20h) [reset = 0h]

OUTPUT1MUXENABLE is shown in [Figure 7-38](#) and described in [Table 7-48](#).

Output X-BAR Mux Enable for Output 1

Figure 7-38. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-48. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-48. OUTPUT1MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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>
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>
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>
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>
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>
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>
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.</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.</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.</p>

Table 7-48. OUTPUT1MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-48. OUTPUT1MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.18 OUTPUT2MUXENABLE Register (Offset = 22h) [reset = 0h]

OUTPUT2MUXENABLE is shown in [Figure 7-39](#) and described in [Table 7-49](#).

Output X-BAR Mux Enable for Output 2

Figure 7-39. OUTPUT2MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-49. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-49. OUTPUT2MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-49. OUTPUT2MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-49. OUTPUT2MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.19 OUTPUT3MUXENABLE Register (Offset = 24h) [reset = 0h]

OUTPUT3MUXENABLE is shown in [Figure 7-40](#) and described in [Table 7-50](#).

Output X-BAR Mux Enable for Output 3

Figure 7-40. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-50. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-50. OUTPUT3MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-50. OUTPUT3MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-50. OUTPUT3MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.20 OUTPUT4MUXENABLE Register (Offset = 26h) [reset = 0h]

OUTPUT4MUXENABLE is shown in [Figure 7-41](#) and described in [Table 7-51](#).

Output X-BAR Mux Enable for Output 4

Figure 7-41. OUTPUT4MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-51. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-51. OUTPUT4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-51. OUTPUT4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-51. OUTPUT4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.21 OUTPUT5MUXENABLE Register (Offset = 28h) [reset = 0h]

OUTPUT5MUXENABLE is shown in [Figure 7-42](#) and described in [Table 7-52](#).

Output X-BAR Mux Enable for Output 5

Figure 7-42. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-52. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-52. OUTPUT5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-52. OUTPUT5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-52. OUTPUT5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.22 OUTPUT6MUXENABLE Register (Offset = 2Ah) [reset = 0h]

OUTPUT6MUXENABLE is shown in [Figure 7-43](#) and described in [Table 7-53](#).

Output X-BAR Mux Enable for Output 6

Figure 7-43. OUTPUT6MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-53. 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.
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.
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.
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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-53. OUTPUT6MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-53. OUTPUT6MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-53. OUTPUT6MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.23 OUTPUT7MUXENABLE Register (Offset = 2Ch) [reset = 0h]

OUTPUT7MUXENABLE is shown in [Figure 7-44](#) and described in [Table 7-54](#).

Output X-BAR Mux Enable for Output 7

Figure 7-44. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-54. OUTPUT7MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-54. OUTPUT7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-54. OUTPUT7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-54. OUTPUT7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.24 OUTPUT8MUXENABLE Register (Offset = 2Eh) [reset = 0h]

OUTPUT8MUXENABLE is shown in [Figure 7-45](#) and described in [Table 7-55](#).

Output X-BAR Mux Enable for Output 8

Figure 7-45. OUTPUT8MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-55. OUTPUT8MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux27 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux27 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details.

Table 7-55. OUTPUT8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 7-55. OUTPUT8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 7-55. OUTPUT8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

7.11.3.25 OUTPUTLATCH Register (Offset = 30h) [reset = 0h]

OUTPUTLATCH is shown in [Figure 7-46](#) and described in [Table 7-56](#).

Output X-BAR Output Latch

Figure 7-46. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-56. 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 was not 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
6	OUTPUT7	R	0h	Records the OUTPUT7 of OUTPUT-XBAR. 0: Respective output was not 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
5	OUTPUT6	R	0h	Records the OUTPUT6 of OUTPUT-XBAR. 0: Respective output was not 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
4	OUTPUT5	R	0h	Records the OUTPUT5 of OUTPUT-XBAR. 0: Respective output was not 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

Table 7-56. OUTPUTLATCH Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	OUTPUT4	R	0h	Records the OUTPUT4 of OUTPUT-XBAR. 0: Respective output was not 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
2	OUTPUT3	R	0h	Records the OUTPUT3 of OUTPUT-XBAR. 0: Respective output was not 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
1	OUTPUT2	R	0h	Records the OUTPUT2 of OUTPUT-XBAR. 0: Respective output was not 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
0	OUTPUT1	R	0h	Records the OUTPUT1 of OUTPUT-XBAR. 0: Respective output was not 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

7.11.3.26 OUTPUTLATCHCLR Register (Offset = 32h) [reset = 0h]

OUTPUTLATCHCLR is shown in [Figure 7-47](#) and described in [Table 7-57](#).

Output X-BAR Output Latch Clear

Figure 7-47. 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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-57. 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/W=1	0h	Clears the Output-Latch for OUTPUT8 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
6	OUTPUT7	R=0/W=1	0h	Clears the Output-Latch for OUTPUT7 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
5	OUTPUT6	R=0/W=1	0h	Clears the Output-Latch for OUTPUT6 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
4	OUTPUT5	R=0/W=1	0h	Clears the Output-Latch for OUTPUT5 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
3	OUTPUT4	R=0/W=1	0h	Clears the Output-Latch for OUTPUT4 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.

Table 7-57. OUTPUTLATCHCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	OUTPUT3	R=0/W=1	0h	Clears the Output-Latch for OUTPUT3 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
1	OUTPUT2	R=0/W=1	0h	Clears the Output-Latch for OUTPUT2 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
0	OUTPUT1	R=0/W=1	0h	Clears the Output-Latch for OUTPUT1 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.

7.11.3.27 OUTPUTLATCHFRC Register (Offset = 34h) [reset = 0h]

OUTPUTLATCHFRC is shown in [Figure 7-48](#) and described in [Table 7-58](#).

Output X-BAR Output Latch Clear

Figure 7-48. 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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-58. 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/W=1	0h	Sets the Output-Latch for OUTPUT8 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
6	OUTPUT7	R=0/W=1	0h	Sets the Output-Latch for OUTPUT7 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
5	OUTPUT6	R=0/W=1	0h	Sets the Output-Latch for OUTPUT6 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
4	OUTPUT5	R=0/W=1	0h	Sets the Output-Latch for OUTPUT5 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.
3	OUTPUT4	R=0/W=1	0h	Sets the Output-Latch for OUTPUT4 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details.

Table 7-58. OUTPUTLATCHFRC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	OUTPUT3	R=0/W=1	0h	<p>Sets the Output-Latch for OUTPUT3 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT 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>
1	OUTPUT2	R=0/W=1	0h	<p>Sets the Output-Latch for OUTPUT2 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT 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>
0	OUTPUT1	R=0/W=1	0h	<p>Sets the Output-Latch for OUTPUT1 of OUTPUT-XBAR</p> <p>Writing 1 sets the corresponding output latch bit in the OUTPUT-XBAROLAT 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>

7.11.3.28 OUTPUTLATCHENABLE Register (Offset = 36h) [reset = 0h]

OUTPUTLATCHENABLE is shown in [Figure 7-49](#) and described in [Table 7-59](#).

Output X-BAR Output Latch Enable

Figure 7-49. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-59. 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.
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.
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.
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.
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.
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.

Table 7-59. OUTPUTLATCHENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.

7.11.3.29 OUTPUTINV Register (Offset = 38h) [reset = 0h]

OUTPUTINV is shown in [Figure 7-50](#) and described in [Table 7-60](#).

Output X-BAR Output Inversion

Figure 7-50. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-60. 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.
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.
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.
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.
3	OUTPUT4	R/W	0h	Selects polarity for OUTPUT4 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.
2	OUTPUT3	R/W	0h	Selects polarity for OUTPUT3 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.

Table 7-60. OUTPUTINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	OUTPUT2	R/W	0h	Selects polarity for OUTPUT2 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.
0	OUTPUT1	R/W	0h	Selects polarity for OUTPUT1 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.

7.11.3.30 OUTPUTLOCK Register (Offset = 3Eh) [reset = 0h]

OUTPUTLOCK is shown in [Figure 7-51](#) and described in [Table 7-61](#).

Output X-BAR Configuration Lock register

Figure 7-51. OUTPUTLOCK Register

31	30	29	28	27	26	25	24
KEY							
R=0/W=1-0h							
23	22	21	20	19	18	17	16
KEY							
R=0/W=1-0h							
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/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-61. OUTPUTLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	R=0/W=1	0h	Bit-0 of this register can be set only if KEY= 0x5a5a
15-1	RESERVED	R=0	0h	Reserved
0	LOCK	R/SOnce	0h	<p>Locks the configuration for OUTPUT-XBAR. Once the configuration is locked, writes to the below registers for OUTPUT-XBAR is blocked.</p> <p>Registers Affected by the LOCK mechanism:</p> <p>OUTPUT-XBAROUTyMUX0TO15CFG</p> <p>OUTPUT-XBAROUTyMUX16TO31CFG</p> <p>OUTPUT-XBAROUTyMUXENABLE</p> <p>OUTPUT-XBAROUTLATEN</p> <p>OUTPUT-XBAROUTINV</p> <p>0: Writes to the above registers are allowed</p> <p>1: Writes to the above registers are blocked</p> <p>Note:</p> <p>[1] LOCK mechanism only applies to writes. Reads are never blocked.</p>

7.11.4 GPIO_CTRL_REGS Registers

Table 7-62 lists the memory-mapped registers for the GPIO_CTRL_REGS. All register offset addresses not listed in Table 7-62 should be considered as reserved locations and the register contents should not be modified.

Table 7-62. GPIO_CTRL_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	GPACTRL	GPIO A Qualification Sampling Period Control (GPIO0 to 31)	EALLOW	Go
2h	GPAQSEL1	GPIO A Qualifier Select 1 Register (GPIO0 to 15)	EALLOW	Go
4h	GPAQSEL2	GPIO A Qualifier Select 2 Register (GPIO16 to 31)	EALLOW	Go
6h	GPAMUX1	GPIO A Mux 1 Register (GPIO0 to 15)	EALLOW	Go
8h	GPAMUX2	GPIO A Mux 2 Register (GPIO16 to 31)	EALLOW	Go
Ah	GPADIR	GPIO A Direction Register (GPIO0 to 31)	EALLOW	Go
Ch	GPAPUD	GPIO A Pull Up Disable Register (GPIO0 to 31)	EALLOW	Go
10h	GPAINV	GPIO A Input Polarity Invert Registers (GPIO0 to 31)	EALLOW	Go
12h	GPAODR	GPIO A Open Drain Output Register (GPIO0 to GPIO31)	EALLOW	Go
20h	GPAGMUX1	GPIO A Peripheral Group Mux (GPIO0 to 15)	EALLOW	Go
22h	GPAGMUX2	GPIO A Peripheral Group Mux (GPIO16 to 31)	EALLOW	Go
28h	GPACSEL1	GPIO A Core Select Register (GPIO0 to 7)	EALLOW	Go
2Ah	GPACSEL2	GPIO A Core Select Register (GPIO8 to 15)	EALLOW	Go
2Ch	GPACSEL3	GPIO A Core Select Register (GPIO16 to 23)	EALLOW	Go
2Eh	GPACSEL4	GPIO A Core Select Register (GPIO24 to 31)	EALLOW	Go
3Ch	GPALOCK	GPIO A Lock Configuration Register (GPIO0 to 31)	EALLOW	Go
3Eh	GPACR	GPIO A Lock Commit Register (GPIO0 to 31)	EALLOW	Go
40h	GPBCTRL	GPIO B Qualification Sampling Period Control (GPIO32 to 63)	EALLOW	Go
42h	GPBQSEL1	GPIO B Qualifier Select 1 Register (GPIO32 to 47)	EALLOW	Go
44h	GPBQSEL2	GPIO B Qualifier Select 2 Register (GPIO48 to 63)	EALLOW	Go
46h	GPBMUX1	GPIO B Mux 1 Register (GPIO32 to 47)	EALLOW	Go
48h	GPBMUX2	GPIO B Mux 2 Register (GPIO48 to 63)	EALLOW	Go
4Ah	GPBDIR	GPIO B Direction Register (GPIO32 to 63)	EALLOW	Go
4Ch	GPBPUD	GPIO B Pull Up Disable Register (GPIO32 to 63)	EALLOW	Go
50h	GPBINV	GPIO B Input Polarity Invert Registers (GPIO32 to 63)	EALLOW	Go
52h	GPBODR	GPIO B Open Drain Output Register (GPIO32 to GPIO63)	EALLOW	Go
54h	GPBAMSEL	GPIO B Analog Mode Select register (GPIO32 to GPIO63)	EALLOW	Go
60h	GPBGMUX1	GPIO B Peripheral Group Mux (GPIO32 to 47)	EALLOW	Go
62h	GPBGMUX2	GPIO B Peripheral Group Mux (GPIO48 to 63)	EALLOW	Go
68h	GPBCSEL1	GPIO B Core Select Register (GPIO32 to 39)	EALLOW	Go
6Ah	GPBCSEL2	GPIO B Core Select Register (GPIO40 to 47)	EALLOW	Go
6Ch	GPBCSEL3	GPIO B Core Select Register (GPIO48 to 55)	EALLOW	Go
6Eh	GPBCSEL4	GPIO B Core Select Register (GPIO56 to 63)	EALLOW	Go
7Ch	GPBLOCK	GPIO B Lock Configuration Register (GPIO32 to 63)	EALLOW	Go

Table 7-62. GPIO_CTRL_REGS Registers (continued)

Offset	Acronym	Register Name	Write Protection	Section
7Eh	GPBCR	GPIO B Lock Commit Register (GPIO32 to 63)	EALLOW	Go
80h	GPCCTRL	GPIO C Qualification Sampling Period Control (GPIO64 to 95)	EALLOW	Go
82h	GPCQSEL1	GPIO C Qualifier Select 1 Register (GPIO64 to 79)	EALLOW	Go
84h	GPCQSEL2	GPIO C Qualifier Select 2 Register (GPIO80 to 95)	EALLOW	Go
86h	GPCMUX1	GPIO C Mux 1 Register (GPIO64 to 79)	EALLOW	Go
88h	GPCMUX2	GPIO C Mux 2 Register (GPIO80 to 95)	EALLOW	Go
8Ah	GPCDIR	GPIO C Direction Register (GPIO64 to 95)	EALLOW	Go
8Ch	GPCPUD	GPIO C Pull Up Disable Register (GPIO64 to 95)	EALLOW	Go
90h	GPCINV	GPIO C Input Polarity Invert Registers (GPIO64 to 95)	EALLOW	Go
92h	GPCODR	GPIO C Open Drain Output Register (GPIO64 to GPIO95)	EALLOW	Go
A0h	GPCGMUX1	GPIO C Peripheral Group Mux (GPIO64 to 79)	EALLOW	Go
A2h	GPCGMUX2	GPIO C Peripheral Group Mux (GPIO80 to 95)	EALLOW	Go
A8h	GPCCSEL1	GPIO C Core Select Register (GPIO64 to 71)	EALLOW	Go
AAh	GPCCSEL2	GPIO C Core Select Register (GPIO72 to 79)	EALLOW	Go
ACh	GPCCSEL3	GPIO C Core Select Register (GPIO80 to 87)	EALLOW	Go
AEh	GPCCSEL4	GPIO C Core Select Register (GPIO88 to 95)	EALLOW	Go
BCh	GPCLOCK	GPIO C Lock Configuration Register (GPIO64 to 95)	EALLOW	Go
BEh	GPCCR	GPIO C Lock Commit Register (GPIO64 to 95)	EALLOW	Go
C0h	GPDCTRL	GPIO D Qualification Sampling Period Control (GPIO96 to 127)	EALLOW	Go
C2h	GPDQSEL1	GPIO D Qualifier Select 1 Register (GPIO96 to 111)	EALLOW	Go
C4h	GPDQSEL2	GPIO D Qualifier Select 2 Register (GPIO112 to 127)	EALLOW	Go
C6h	GPDMUX1	GPIO D Mux 1 Register (GPIO96 to 111)	EALLOW	Go
C8h	GPDMUX2	GPIO D Mux 2 Register (GPIO112 to 127)	EALLOW	Go
CAh	GPDDIR	GPIO D Direction Register (GPIO96 to 127)	EALLOW	Go
CCh	GPDPUD	GPIO D Pull Up Disable Register (GPIO96 to 127)	EALLOW	Go
D0h	GPDINV	GPIO D Input Polarity Invert Registers (GPIO96 to 127)	EALLOW	Go
D2h	GPDODR	GPIO D Open Drain Output Register (GPIO96 to GPIO127)	EALLOW	Go
E0h	GPDGMUX1	GPIO D Peripheral Group Mux (GPIO96 to 111)	EALLOW	Go
E2h	GPDGMUX2	GPIO D Peripheral Group Mux (GPIO112 to 127)	EALLOW	Go
E8h	GPDCSEL1	GPIO D Core Select Register (GPIO96 to 103)	EALLOW	Go
EAh	GPDCSEL2	GPIO D Core Select Register (GPIO104 to 111)	EALLOW	Go
ECh	GPDCSEL3	GPIO D Core Select Register (GPIO112 to 119)	EALLOW	Go
EEh	GPDCSEL4	GPIO D Core Select Register (GPIO120 to 127)	EALLOW	Go
FCh	GPDLCK	GPIO D Lock Configuration Register (GPIO96 to 127)	EALLOW	Go
FEh	GPDCR	GPIO D Lock Commit Register (GPIO96 to 127)	EALLOW	Go
100h	GPECTRL	GPIO E Qualification Sampling Period Control (GPIO128 to 159)	EALLOW	Go
102h	GPEQSEL1	GPIO E Qualifier Select 1 Register (GPIO128 to 143)	EALLOW	Go

Table 7-62. GPIO_CTRL_REGS Registers (continued)

Offset	Acronym	Register Name	Write Protection	Section
104h	GPEQSEL2	GPIO E Qualifier Select 2 Register (GPIO144 to 159)	EALLOW	Go
106h	GPEMUX1	GPIO E Mux 1 Register (GPIO128 to 143)	EALLOW	Go
108h	GPEMUX2	GPIO E Mux 2 Register (GPIO144 to 159)	EALLOW	Go
10Ah	GPEDIR	GPIO E Direction Register (GPIO128 to 159)	EALLOW	Go
10Ch	GPEPUD	GPIO E Pull Up Disable Register (GPIO128 to 159)	EALLOW	Go
110h	GPEINV	GPIO E Input Polarity Invert Registers (GPIO128 to 159)	EALLOW	Go
112h	GPEODR	GPIO E Open Drain Output Register (GPIO128 to GPIO159)	EALLOW	Go
120h	GPEGMUX1	GPIO E Peripheral Group Mux (GPIO128 to 143)	EALLOW	Go
122h	GPEGMUX2	GPIO E Peripheral Group Mux (GPIO144 to 159)	EALLOW	Go
128h	GPECSEL1	GPIO E Core Select Register (GPIO128 to 135)	EALLOW	Go
12Ah	GPECSEL2	GPIO E Core Select Register (GPIO136 to 143)	EALLOW	Go
12Ch	GPECSEL3	GPIO E Core Select Register (GPIO144 to 151)	EALLOW	Go
12Eh	GPECSEL4	GPIO E Core Select Register (GPIO152 to 159)	EALLOW	Go
13Ch	GPELOCK	GPIO E Lock Configuration Register (GPIO128 to 159)	EALLOW	Go
13Eh	GPECR	GPIO E Lock Commit Register (GPIO128 to 159)	EALLOW	Go
140h	GPFCTRL	GPIO F Qualification Sampling Period Control (GPIO160 to 168)	EALLOW	Go
142h	GPFQSEL1	GPIO F Qualifier Select 1 Register (GPIO160 to 168)	EALLOW	Go
146h	GPFMUX1	GPIO F Mux 1 Register (GPIO160 to 168)	EALLOW	Go
14Ah	GPFDIR	GPIO F Direction Register (GPIO160 to 168)	EALLOW	Go
14Ch	GPFPUD	GPIO F Pull Up Disable Register (GPIO160 to 168)	EALLOW	Go
150h	GPFINV	GPIO F Input Polarity Invert Registers (GPIO160 to 168)	EALLOW	Go
152h	GPFODR	GPIO F Open Drain Output Register (GPIO160 to GPIO168)	EALLOW	Go
160h	GPFGMUX1	GPIO F Peripheral Group Mux (GPIO160 to 168)	EALLOW	Go
168h	GPFCSEL1	GPIO F Core Select Register (GPIO160 to 167)	EALLOW	Go
16Ah	GPFCSEL2	GPIO F Core Select Register (GPIO168)	EALLOW	Go
17Ch	GPFLLOCK	GPIO F Lock Configuration Register (GPIO160 to 168)	EALLOW	Go
17Eh	GPFCR	GPIO F Lock Commit Register (GPIO160 to 168)	EALLOW	Go

7.11.4.1 GPACTRL Register (Offset = 0h) [reset = 0h]

GPACTRL is shown in [Figure 7-52](#) and described in [Table 7-63](#).

GPIO A Qualification Sampling Period Control (GPIO0 to 31)

Figure 7-52. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-63. GPACTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO24 to GPIO31: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO16 to GPIO23: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO8 to GPIO15: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO0 to GPIO7: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510

7.11.4.2 GPAQSEL1 Register (Offset = 2h) [reset = 0h]

GPAQSEL1 is shown in [Figure 7-53](#) and described in [Table 7-64](#).

GPIO A Qualifier Select 1 Register (GPIO0 to 15)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-53. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-64. GPAQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Input qualification type
29-28	GPIO14	R/W	0h	Input qualification type
27-26	GPIO13	R/W	0h	Input qualification type
25-24	GPIO12	R/W	0h	Input qualification type
23-22	GPIO11	R/W	0h	Input qualification type
21-20	GPIO10	R/W	0h	Input qualification type
19-18	GPIO9	R/W	0h	Input qualification type
17-16	GPIO8	R/W	0h	Input qualification type
15-14	GPIO7	R/W	0h	Input qualification type
13-12	GPIO6	R/W	0h	Input qualification type
11-10	GPIO5	R/W	0h	Input qualification type
9-8	GPIO4	R/W	0h	Input qualification type
7-6	GPIO3	R/W	0h	Input qualification type
5-4	GPIO2	R/W	0h	Input qualification type
3-2	GPIO1	R/W	0h	Input qualification type
1-0	GPIO0	R/W	0h	Input qualification type

7.11.4.3 GPAQSEL2 Register (Offset = 4h) [reset = 0h]

GPAQSEL2 is shown in [Figure 7-54](#) and described in [Table 7-65](#).

GPIO A Qualifier Select 2 Register (GPIO16 to 31)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-54. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-65. GPAQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Input qualification type
29-28	GPIO30	R/W	0h	Input qualification type
27-26	GPIO29	R/W	0h	Input qualification type
25-24	GPIO28	R/W	0h	Input qualification type
23-22	GPIO27	R/W	0h	Input qualification type
21-20	GPIO26	R/W	0h	Input qualification type
19-18	GPIO25	R/W	0h	Input qualification type
17-16	GPIO24	R/W	0h	Input qualification type
15-14	GPIO23	R/W	0h	Input qualification type
13-12	GPIO22	R/W	0h	Input qualification type
11-10	GPIO21	R/W	0h	Input qualification type
9-8	GPIO20	R/W	0h	Input qualification type
7-6	GPIO19	R/W	0h	Input qualification type
5-4	GPIO18	R/W	0h	Input qualification type
3-2	GPIO17	R/W	0h	Input qualification type
1-0	GPIO16	R/W	0h	Input qualification type

7.11.4.4 GPAMUX1 Register (Offset = 6h) [reset = 0h]

GPAMUX1 is shown in [Figure 7-55](#) and described in [Table 7-66](#).

GPIO A Mux 1 Register (GPIO0 to 15)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-55. 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								

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-66. GPAMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO14	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO13	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO12	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO11	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO10	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO9	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO8	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO7	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO6	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO5	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO4	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO3	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO2	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO1	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO0	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.5 GPAMUX2 Register (Offset = 8h) [reset = 0h]

GPAMUX2 is shown in [Figure 7-56](#) and described in [Table 7-67](#).

GPIO A Mux 2 Register (GPIO16 to 31)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-56. 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								

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-67. GPAMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO30	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO29	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO28	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO27	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO26	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO25	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO24	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO23	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO22	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO21	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO20	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO19	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO18	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO17	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO16	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.6 GPADIR Register (Offset = Ah) [reset = 0h]

GPADIR is shown in [Figure 7-57](#) and described in [Table 7-68](#).

GPIO A Direction Register (GPIO0 to 31)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-57. GPADIR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-68. GPADIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Defines direction for this pin in GPIO mode
30	GPIO30	R/W	0h	Defines direction for this pin in GPIO mode
29	GPIO29	R/W	0h	Defines direction for this pin in GPIO mode
28	GPIO28	R/W	0h	Defines direction for this pin in GPIO mode
27	GPIO27	R/W	0h	Defines direction for this pin in GPIO mode
26	GPIO26	R/W	0h	Defines direction for this pin in GPIO mode
25	GPIO25	R/W	0h	Defines direction for this pin in GPIO mode
24	GPIO24	R/W	0h	Defines direction for this pin in GPIO mode
23	GPIO23	R/W	0h	Defines direction for this pin in GPIO mode
22	GPIO22	R/W	0h	Defines direction for this pin in GPIO mode
21	GPIO21	R/W	0h	Defines direction for this pin in GPIO mode
20	GPIO20	R/W	0h	Defines direction for this pin in GPIO mode
19	GPIO19	R/W	0h	Defines direction for this pin in GPIO mode
18	GPIO18	R/W	0h	Defines direction for this pin in GPIO mode
17	GPIO17	R/W	0h	Defines direction for this pin in GPIO mode
16	GPIO16	R/W	0h	Defines direction for this pin in GPIO mode
15	GPIO15	R/W	0h	Defines direction for this pin in GPIO mode
14	GPIO14	R/W	0h	Defines direction for this pin in GPIO mode
13	GPIO13	R/W	0h	Defines direction for this pin in GPIO mode
12	GPIO12	R/W	0h	Defines direction for this pin in GPIO mode
11	GPIO11	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-68. GPADIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO10	R/W	0h	Defines direction for this pin in GPIO mode
9	GPIO9	R/W	0h	Defines direction for this pin in GPIO mode
8	GPIO8	R/W	0h	Defines direction for this pin in GPIO mode
7	GPIO7	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO6	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO5	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO4	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO3	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO2	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO1	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO0	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.7 GPAPUD Register (Offset = Ch) [reset = FFFFFFFFh]

GPAPUD is shown in [Figure 7-58](#) and described in [Table 7-69](#).

GPIO A Pull Up Disable Register (GPIO0 to 31)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-58. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-69. GPAPUD Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	1h	Pull-Up Disable control for this pin
30	GPIO30	R/W	1h	Pull-Up Disable control for this pin
29	GPIO29	R/W	1h	Pull-Up Disable control for this pin
28	GPIO28	R/W	1h	Pull-Up Disable control for this pin
27	GPIO27	R/W	1h	Pull-Up Disable control for this pin
26	GPIO26	R/W	1h	Pull-Up Disable control for this pin
25	GPIO25	R/W	1h	Pull-Up Disable control for this pin
24	GPIO24	R/W	1h	Pull-Up Disable control for this pin
23	GPIO23	R/W	1h	Pull-Up Disable control for this pin
22	GPIO22	R/W	1h	Pull-Up Disable control for this pin
21	GPIO21	R/W	1h	Pull-Up Disable control for this pin
20	GPIO20	R/W	1h	Pull-Up Disable control for this pin
19	GPIO19	R/W	1h	Pull-Up Disable control for this pin
18	GPIO18	R/W	1h	Pull-Up Disable control for this pin
17	GPIO17	R/W	1h	Pull-Up Disable control for this pin
16	GPIO16	R/W	1h	Pull-Up Disable control for this pin
15	GPIO15	R/W	1h	Pull-Up Disable control for this pin
14	GPIO14	R/W	1h	Pull-Up Disable control for this pin
13	GPIO13	R/W	1h	Pull-Up Disable control for this pin
12	GPIO12	R/W	1h	Pull-Up Disable control for this pin
11	GPIO11	R/W	1h	Pull-Up Disable control for this pin

Table 7-69. GPAPUD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO10	R/W	1h	Pull-Up Disable control for this pin
9	GPIO9	R/W	1h	Pull-Up Disable control for this pin
8	GPIO8	R/W	1h	Pull-Up Disable control for this pin
7	GPIO7	R/W	1h	Pull-Up Disable control for this pin
6	GPIO6	R/W	1h	Pull-Up Disable control for this pin
5	GPIO5	R/W	1h	Pull-Up Disable control for this pin
4	GPIO4	R/W	1h	Pull-Up Disable control for this pin
3	GPIO3	R/W	1h	Pull-Up Disable control for this pin
2	GPIO2	R/W	1h	Pull-Up Disable control for this pin
1	GPIO1	R/W	1h	Pull-Up Disable control for this pin
0	GPIO0	R/W	1h	Pull-Up Disable control for this pin

7.11.4.8 GPAINV Register (Offset = 10h) [reset = 0h]

GPAINV is shown in [Figure 7-59](#) and described in [Table 7-70](#).

GPIO A Input Polarity Invert Registers (GPIO0 to 31)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-59. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-70. GPAINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Input inversion control for this pin
30	GPIO30	R/W	0h	Input inversion control for this pin
29	GPIO29	R/W	0h	Input inversion control for this pin
28	GPIO28	R/W	0h	Input inversion control for this pin
27	GPIO27	R/W	0h	Input inversion control for this pin
26	GPIO26	R/W	0h	Input inversion control for this pin
25	GPIO25	R/W	0h	Input inversion control for this pin
24	GPIO24	R/W	0h	Input inversion control for this pin
23	GPIO23	R/W	0h	Input inversion control for this pin
22	GPIO22	R/W	0h	Input inversion control for this pin
21	GPIO21	R/W	0h	Input inversion control for this pin
20	GPIO20	R/W	0h	Input inversion control for this pin
19	GPIO19	R/W	0h	Input inversion control for this pin
18	GPIO18	R/W	0h	Input inversion control for this pin
17	GPIO17	R/W	0h	Input inversion control for this pin
16	GPIO16	R/W	0h	Input inversion control for this pin
15	GPIO15	R/W	0h	Input inversion control for this pin
14	GPIO14	R/W	0h	Input inversion control for this pin
13	GPIO13	R/W	0h	Input inversion control for this pin
12	GPIO12	R/W	0h	Input inversion control for this pin
11	GPIO11	R/W	0h	Input inversion control for this pin

Table 7-70. GPAINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO10	R/W	0h	Input inversion control for this pin
9	GPIO9	R/W	0h	Input inversion control for this pin
8	GPIO8	R/W	0h	Input inversion control for this pin
7	GPIO7	R/W	0h	Input inversion control for this pin
6	GPIO6	R/W	0h	Input inversion control for this pin
5	GPIO5	R/W	0h	Input inversion control for this pin
4	GPIO4	R/W	0h	Input inversion control for this pin
3	GPIO3	R/W	0h	Input inversion control for this pin
2	GPIO2	R/W	0h	Input inversion control for this pin
1	GPIO1	R/W	0h	Input inversion control for this pin
0	GPIO0	R/W	0h	Input inversion control for this pin

7.11.4.9 GPAODR Register (Offset = 12h) [reset = 0h]

GPAODR is shown in [Figure 7-60](#) and described in [Table 7-71](#).

GPIO A Open Drain Output Register (GPIO0 to GPIO31)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-60. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-71. GPAODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output Open-Drain control for this pin
30	GPIO30	R/W	0h	Output Open-Drain control for this pin
29	GPIO29	R/W	0h	Output Open-Drain control for this pin
28	GPIO28	R/W	0h	Output Open-Drain control for this pin
27	GPIO27	R/W	0h	Output Open-Drain control for this pin
26	GPIO26	R/W	0h	Output Open-Drain control for this pin
25	GPIO25	R/W	0h	Output Open-Drain control for this pin
24	GPIO24	R/W	0h	Output Open-Drain control for this pin
23	GPIO23	R/W	0h	Output Open-Drain control for this pin
22	GPIO22	R/W	0h	Output Open-Drain control for this pin
21	GPIO21	R/W	0h	Output Open-Drain control for this pin
20	GPIO20	R/W	0h	Output Open-Drain control for this pin
19	GPIO19	R/W	0h	Output Open-Drain control for this pin
18	GPIO18	R/W	0h	Output Open-Drain control for this pin
17	GPIO17	R/W	0h	Output Open-Drain control for this pin
16	GPIO16	R/W	0h	Output Open-Drain control for this pin
15	GPIO15	R/W	0h	Output Open-Drain control for this pin
14	GPIO14	R/W	0h	Output Open-Drain control for this pin
13	GPIO13	R/W	0h	Output Open-Drain control for this pin

Table 7-71. GPAODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	GPIO12	R/W	0h	Output Open-Drain control for this pin
11	GPIO11	R/W	0h	Output Open-Drain control for this pin
10	GPIO10	R/W	0h	Output Open-Drain control for this pin
9	GPIO9	R/W	0h	Output Open-Drain control for this pin
8	GPIO8	R/W	0h	Output Open-Drain control for this pin
7	GPIO7	R/W	0h	Output Open-Drain control for this pin
6	GPIO6	R/W	0h	Output Open-Drain control for this pin
5	GPIO5	R/W	0h	Output Open-Drain control for this pin
4	GPIO4	R/W	0h	Output Open-Drain control for this pin
3	GPIO3	R/W	0h	Output Open-Drain control for this pin
2	GPIO2	R/W	0h	Output Open-Drain control for this pin
1	GPIO1	R/W	0h	Output Open-Drain control for this pin
0	GPIO0	R/W	0h	Output Open-Drain control for this pin

7.11.4.10 GPAGMUX1 Register (Offset = 20h) [reset = 0h]

GPAGMUX1 is shown in [Figure 7-61](#) and described in [Table 7-72](#).

GPIO A Peripheral Group Mux (GPIO0 to 15)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-61. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-72. GPAGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO15	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO14	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO13	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO12	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO11	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO10	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO9	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO8	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO7	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO6	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO5	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO4	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO3	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO2	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO1	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO0	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.11 GPAGMUX2 Register (Offset = 22h) [reset = 0h]

GPAGMUX2 is shown in [Figure 7-62](#) and described in [Table 7-73](#).

GPIO A Peripheral Group Mux (GPIO16 to 31)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-62. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-73. GPAGMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO31	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO30	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO29	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO28	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO27	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO26	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO25	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO24	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO23	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO22	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO21	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO20	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO19	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO18	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO17	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO16	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.12 GPACSEL1 Register (Offset = 28h) [reset = 0h]

GPACSEL1 is shown in [Figure 7-63](#) and described in [Table 7-74](#).

GPIO A Core Select Register (GPIO0 to 7)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-63. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-74. GPACSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO7	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO6	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO5	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO4	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO3	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO2	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO1	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO0	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.13 GPACSEL2 Register (Offset = 2Ah) [reset = 0h]

GPACSEL2 is shown in [Figure 7-64](#) and described in [Table 7-75](#).

GPIO A Core Select Register (GPIO8 to 15)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-64. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-75. GPACSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO15	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO14	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO13	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO12	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO11	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO10	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO9	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO8	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.14 GPACSEL3 Register (Offset = 2Ch) [reset = 0h]

GPACSEL3 is shown in [Figure 7-65](#) and described in [Table 7-76](#).

GPIO A Core Select Register (GPIO16 to 23)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-65. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-76. GPACSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO23	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO22	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO21	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO20	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO19	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO18	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO17	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO16	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.15 GPACSEL4 Register (Offset = 2Eh) [reset = 0h]

GPACSEL4 is shown in [Figure 7-66](#) and described in [Table 7-77](#).

GPIO A Core Select Register (GPIO24 to 31)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-66. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-77. GPACSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO31	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO30	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO29	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO28	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO27	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO26	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO25	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO24	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.16 GPALOCK Register (Offset = 3Ch) [reset = 0h]

GPALOCK is shown in [Figure 7-67](#) and described in [Table 7-78](#).

GPIO A Lock Configuration Register (GPIO0 to 31)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-67. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-78. GPALOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Configuration Lock bit for this pin
30	GPIO30	R/W	0h	Configuration Lock bit for this pin
29	GPIO29	R/W	0h	Configuration Lock bit for this pin
28	GPIO28	R/W	0h	Configuration Lock bit for this pin
27	GPIO27	R/W	0h	Configuration Lock bit for this pin
26	GPIO26	R/W	0h	Configuration Lock bit for this pin
25	GPIO25	R/W	0h	Configuration Lock bit for this pin
24	GPIO24	R/W	0h	Configuration Lock bit for this pin
23	GPIO23	R/W	0h	Configuration Lock bit for this pin
22	GPIO22	R/W	0h	Configuration Lock bit for this pin
21	GPIO21	R/W	0h	Configuration Lock bit for this pin
20	GPIO20	R/W	0h	Configuration Lock bit for this pin
19	GPIO19	R/W	0h	Configuration Lock bit for this pin
18	GPIO18	R/W	0h	Configuration Lock bit for this pin
17	GPIO17	R/W	0h	Configuration Lock bit for this pin
16	GPIO16	R/W	0h	Configuration Lock bit for this pin
15	GPIO15	R/W	0h	Configuration Lock bit for this pin
14	GPIO14	R/W	0h	Configuration Lock bit for this pin
13	GPIO13	R/W	0h	Configuration Lock bit for this pin
12	GPIO12	R/W	0h	Configuration Lock bit for this pin

Table 7-78. GPALOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	GPIO11	R/W	0h	Configuration Lock bit for this pin
10	GPIO10	R/W	0h	Configuration Lock bit for this pin
9	GPIO9	R/W	0h	Configuration Lock bit for this pin
8	GPIO8	R/W	0h	Configuration Lock bit for this pin
7	GPIO7	R/W	0h	Configuration Lock bit for this pin
6	GPIO6	R/W	0h	Configuration Lock bit for this pin
5	GPIO5	R/W	0h	Configuration Lock bit for this pin
4	GPIO4	R/W	0h	Configuration Lock bit for this pin
3	GPIO3	R/W	0h	Configuration Lock bit for this pin
2	GPIO2	R/W	0h	Configuration Lock bit for this pin
1	GPIO1	R/W	0h	Configuration Lock bit for this pin
0	GPIO0	R/W	0h	Configuration Lock bit for this pin

7.11.4.17 GPACR Register (Offset = 3Eh) [reset = 0h]

GPACR is shown in [Figure 7-68](#) and described in [Table 7-79](#).

GPIO A Lock Commit Register (GPIO0 to 31)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-68. GPACR Register

31	30	29	28	27	26	25	24
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
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
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
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
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
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-79. GPACR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/WOnce	0h	Configuration lock commit bit for this pin
30	GPIO30	R/WOnce	0h	Configuration lock commit bit for this pin
29	GPIO29	R/WOnce	0h	Configuration lock commit bit for this pin
28	GPIO28	R/WOnce	0h	Configuration lock commit bit for this pin
27	GPIO27	R/WOnce	0h	Configuration lock commit bit for this pin
26	GPIO26	R/WOnce	0h	Configuration lock commit bit for this pin
25	GPIO25	R/WOnce	0h	Configuration lock commit bit for this pin
24	GPIO24	R/WOnce	0h	Configuration lock commit bit for this pin
23	GPIO23	R/WOnce	0h	Configuration lock commit bit for this pin
22	GPIO22	R/WOnce	0h	Configuration lock commit bit for this pin
21	GPIO21	R/WOnce	0h	Configuration lock commit bit for this pin
20	GPIO20	R/WOnce	0h	Configuration lock commit bit for this pin
19	GPIO19	R/WOnce	0h	Configuration lock commit bit for this pin
18	GPIO18	R/WOnce	0h	Configuration lock commit bit for this pin
17	GPIO17	R/WOnce	0h	Configuration lock commit bit for this pin
16	GPIO16	R/WOnce	0h	Configuration lock commit bit for this pin
15	GPIO15	R/WOnce	0h	Configuration lock commit bit for this pin
14	GPIO14	R/WOnce	0h	Configuration lock commit bit for this pin
13	GPIO13	R/WOnce	0h	Configuration lock commit bit for this pin
12	GPIO12	R/WOnce	0h	Configuration lock commit bit for this pin
11	GPIO11	R/WOnce	0h	Configuration lock commit bit for this pin
10	GPIO10	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-79. GPACR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO9	R/WOnce	0h	Configuration lock commit bit for this pin
8	GPIO8	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO7	R/WOnce	0h	Configuration lock commit bit for this pin
6	GPIO6	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO5	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO4	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO3	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO2	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO1	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO0	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.4.18 GPBCTRL Register (Offset = 40h) [reset = 0h]

GPBCTRL is shown in [Figure 7-69](#) and described in [Table 7-80](#).

GPIO B Qualification Sampling Period Control (GPIO32 to 63)

Figure 7-69. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-80. GPBCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO56 to GPIO63: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO48 to GPIO55: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO40 to GPIO47: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO32 to GPIO39: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510

7.11.4.19 GPBQSEL1 Register (Offset = 42h) [reset = 0h]

GPBQSEL1 is shown in [Figure 7-70](#) and described in [Table 7-81](#).

GPIO B Qualifier Select 1 Register (GPIO32 to 47)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-70. 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		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-81. GPBQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Input qualification type
29-28	GPIO46	R/W	0h	Input qualification type
27-26	GPIO45	R/W	0h	Input qualification type
25-24	GPIO44	R/W	0h	Input qualification type
23-22	GPIO43	R/W	0h	Input qualification type
21-20	GPIO42	R/W	0h	Input qualification type
19-18	GPIO41	R/W	0h	Input qualification type
17-16	GPIO40	R/W	0h	Input qualification type
15-14	GPIO39	R/W	0h	Input qualification type
13-12	GPIO38	R/W	0h	Input qualification type
11-10	GPIO37	R/W	0h	Input qualification type
9-8	GPIO36	R/W	0h	Input qualification type
7-6	GPIO35	R/W	0h	Input qualification type
5-4	GPIO34	R/W	0h	Input qualification type
3-2	GPIO33	R/W	0h	Input qualification type
1-0	GPIO32	R/W	0h	Input qualification type

7.11.4.20 GPBQSEL2 Register (Offset = 44h) [reset = 0h]

GPBQSEL2 is shown in [Figure 7-71](#) and described in [Table 7-82](#).

GPIO B Qualifier Select 2 Register (GPIO48 to 63)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-71. GPBQSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	
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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-82. GPBQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO63	R/W	0h	Input qualification type
29-28	GPIO62	R/W	0h	Input qualification type
27-26	GPIO61	R/W	0h	Input qualification type
25-24	GPIO60	R/W	0h	Input qualification type
23-22	GPIO59	R/W	0h	Input qualification type
21-20	GPIO58	R/W	0h	Input qualification type
19-18	GPIO57	R/W	0h	Input qualification type
17-16	GPIO56	R/W	0h	Input qualification type
15-14	GPIO55	R/W	0h	Input qualification type
13-12	GPIO54	R/W	0h	Input qualification type
11-10	GPIO53	R/W	0h	Input qualification type
9-8	GPIO52	R/W	0h	Input qualification type
7-6	GPIO51	R/W	0h	Input qualification type
5-4	GPIO50	R/W	0h	Input qualification type
3-2	GPIO49	R/W	0h	Input qualification type
1-0	GPIO48	R/W	0h	Input qualification type

7.11.4.21 GPBMUX1 Register (Offset = 46h) [reset = 0h]

GPBMUX1 is shown in [Figure 7-72](#) and described in [Table 7-83](#).

GPIO B Mux 1 Register (GPIO32 to 47)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-72. 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		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-83. GPBMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO46	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO45	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO44	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO43	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO42	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO41	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO40	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO39	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO38	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO37	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO36	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO35	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO34	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO33	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO32	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.22 GPBMUX2 Register (Offset = 48h) [reset = 0h]

GPBMUX2 is shown in [Figure 7-73](#) and described in [Table 7-84](#).

GPIO B Mux 2 Register (GPIO48 to 63)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-73. GPBMUX2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	
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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-84. GPBMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO63	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO62	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO61	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO60	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO59	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO58	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO57	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO56	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO55	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO54	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO53	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO52	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO51	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO50	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO49	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO48	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.23 GPBDIR Register (Offset = 4Ah) [reset = 0h]

GPBDIR is shown in [Figure 7-74](#) and described in [Table 7-85](#).

GPIO B Direction Register (GPIO32 to 63)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-74. GPBDIR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-85. GPBDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Defines direction for this pin in GPIO mode
30	GPIO62	R/W	0h	Defines direction for this pin in GPIO mode
29	GPIO61	R/W	0h	Defines direction for this pin in GPIO mode
28	GPIO60	R/W	0h	Defines direction for this pin in GPIO mode
27	GPIO59	R/W	0h	Defines direction for this pin in GPIO mode
26	GPIO58	R/W	0h	Defines direction for this pin in GPIO mode
25	GPIO57	R/W	0h	Defines direction for this pin in GPIO mode
24	GPIO56	R/W	0h	Defines direction for this pin in GPIO mode
23	GPIO55	R/W	0h	Defines direction for this pin in GPIO mode
22	GPIO54	R/W	0h	Defines direction for this pin in GPIO mode
21	GPIO53	R/W	0h	Defines direction for this pin in GPIO mode
20	GPIO52	R/W	0h	Defines direction for this pin in GPIO mode
19	GPIO51	R/W	0h	Defines direction for this pin in GPIO mode
18	GPIO50	R/W	0h	Defines direction for this pin in GPIO mode
17	GPIO49	R/W	0h	Defines direction for this pin in GPIO mode
16	GPIO48	R/W	0h	Defines direction for this pin in GPIO mode
15	GPIO47	R/W	0h	Defines direction for this pin in GPIO mode
14	GPIO46	R/W	0h	Defines direction for this pin in GPIO mode
13	GPIO45	R/W	0h	Defines direction for this pin in GPIO mode
12	GPIO44	R/W	0h	Defines direction for this pin in GPIO mode
11	GPIO43	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-85. GPBDIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO42	R/W	0h	Defines direction for this pin in GPIO mode
9	GPIO41	R/W	0h	Defines direction for this pin in GPIO mode
8	GPIO40	R/W	0h	Defines direction for this pin in GPIO mode
7	GPIO39	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO38	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO37	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO36	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO35	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO34	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO33	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO32	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.24 GPBPUD Register (Offset = 4Ch) [reset = FFFFFFFFh]

GPBPUD is shown in [Figure 7-75](#) and described in [Table 7-86](#).

GPIO B Pull Up Disable Register (GPIO32 to 63)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-75. GPBPUD Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
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
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
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	GPIO38	GPIO37	GPIO36	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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-86. GPBPUD Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	1h	Pull-Up Disable control for this pin
30	GPIO62	R/W	1h	Pull-Up Disable control for this pin
29	GPIO61	R/W	1h	Pull-Up Disable control for this pin
28	GPIO60	R/W	1h	Pull-Up Disable control for this pin
27	GPIO59	R/W	1h	Pull-Up Disable control for this pin
26	GPIO58	R/W	1h	Pull-Up Disable control for this pin
25	GPIO57	R/W	1h	Pull-Up Disable control for this pin
24	GPIO56	R/W	1h	Pull-Up Disable control for this pin
23	GPIO55	R/W	1h	Pull-Up Disable control for this pin
22	GPIO54	R/W	1h	Pull-Up Disable control for this pin
21	GPIO53	R/W	1h	Pull-Up Disable control for this pin
20	GPIO52	R/W	1h	Pull-Up Disable control for this pin
19	GPIO51	R/W	1h	Pull-Up Disable control for this pin
18	GPIO50	R/W	1h	Pull-Up Disable control for this pin
17	GPIO49	R/W	1h	Pull-Up Disable control for this pin
16	GPIO48	R/W	1h	Pull-Up Disable control for this pin
15	GPIO47	R/W	1h	Pull-Up Disable control for this pin
14	GPIO46	R/W	1h	Pull-Up Disable control for this pin
13	GPIO45	R/W	1h	Pull-Up Disable control for this pin
12	GPIO44	R/W	1h	Pull-Up Disable control for this pin
11	GPIO43	R/W	1h	Pull-Up Disable control for this pin

Table 7-86. GPBPUD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO42	R/W	1h	Pull-Up Disable control for this pin
9	GPIO41	R/W	1h	Pull-Up Disable control for this pin
8	GPIO40	R/W	1h	Pull-Up Disable control for this pin
7	GPIO39	R/W	1h	Pull-Up Disable control for this pin
6	GPIO38	R/W	1h	Pull-Up Disable control for this pin
5	GPIO37	R/W	1h	Pull-Up Disable control for this pin
4	GPIO36	R/W	1h	Pull-Up Disable control for this pin
3	GPIO35	R/W	1h	Pull-Up Disable control for this pin
2	GPIO34	R/W	1h	Pull-Up Disable control for this pin
1	GPIO33	R/W	1h	Pull-Up Disable control for this pin
0	GPIO32	R/W	1h	Pull-Up Disable control for this pin

7.11.4.25 GPBINV Register (Offset = 50h) [reset = 0h]

GPBINV is shown in [Figure 7-76](#) and described in [Table 7-87](#).

GPIO B Input Polarity Invert Registers (GPIO32 to 63)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-76. GPBINV 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-87. GPBINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Input inversion control for this pin
30	GPIO62	R/W	0h	Input inversion control for this pin
29	GPIO61	R/W	0h	Input inversion control for this pin
28	GPIO60	R/W	0h	Input inversion control for this pin
27	GPIO59	R/W	0h	Input inversion control for this pin
26	GPIO58	R/W	0h	Input inversion control for this pin
25	GPIO57	R/W	0h	Input inversion control for this pin
24	GPIO56	R/W	0h	Input inversion control for this pin
23	GPIO55	R/W	0h	Input inversion control for this pin
22	GPIO54	R/W	0h	Input inversion control for this pin
21	GPIO53	R/W	0h	Input inversion control for this pin
20	GPIO52	R/W	0h	Input inversion control for this pin
19	GPIO51	R/W	0h	Input inversion control for this pin
18	GPIO50	R/W	0h	Input inversion control for this pin
17	GPIO49	R/W	0h	Input inversion control for this pin
16	GPIO48	R/W	0h	Input inversion control for this pin
15	GPIO47	R/W	0h	Input inversion control for this pin
14	GPIO46	R/W	0h	Input inversion control for this pin
13	GPIO45	R/W	0h	Input inversion control for this pin
12	GPIO44	R/W	0h	Input inversion control for this pin
11	GPIO43	R/W	0h	Input inversion control for this pin

Table 7-87. GPBINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO42	R/W	0h	Input inversion control for this pin
9	GPIO41	R/W	0h	Input inversion control for this pin
8	GPIO40	R/W	0h	Input inversion control for this pin
7	GPIO39	R/W	0h	Input inversion control for this pin
6	GPIO38	R/W	0h	Input inversion control for this pin
5	GPIO37	R/W	0h	Input inversion control for this pin
4	GPIO36	R/W	0h	Input inversion control for this pin
3	GPIO35	R/W	0h	Input inversion control for this pin
2	GPIO34	R/W	0h	Input inversion control for this pin
1	GPIO33	R/W	0h	Input inversion control for this pin
0	GPIO32	R/W	0h	Input inversion control for this pin

7.11.4.26 GPBODR Register (Offset = 52h) [reset = 0h]

GPBODR is shown in [Figure 7-77](#) and described in [Table 7-88](#).

GPIO B Open Drain Output Register (GPIO32 to GPIO63)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-77. GPBODR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-88. GPBODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Output Open-Drain control for this pin
30	GPIO62	R/W	0h	Output Open-Drain control for this pin
29	GPIO61	R/W	0h	Output Open-Drain control for this pin
28	GPIO60	R/W	0h	Output Open-Drain control for this pin
27	GPIO59	R/W	0h	Output Open-Drain control for this pin
26	GPIO58	R/W	0h	Output Open-Drain control for this pin
25	GPIO57	R/W	0h	Output Open-Drain control for this pin
24	GPIO56	R/W	0h	Output Open-Drain control for this pin
23	GPIO55	R/W	0h	Output Open-Drain control for this pin
22	GPIO54	R/W	0h	Output Open-Drain control for this pin
21	GPIO53	R/W	0h	Output Open-Drain control for this pin
20	GPIO52	R/W	0h	Output Open-Drain control for this pin
19	GPIO51	R/W	0h	Output Open-Drain control for this pin
18	GPIO50	R/W	0h	Output Open-Drain control for this pin
17	GPIO49	R/W	0h	Output Open-Drain control for this pin
16	GPIO48	R/W	0h	Output Open-Drain control for this pin
15	GPIO47	R/W	0h	Output Open-Drain control for this pin
14	GPIO46	R/W	0h	Output Open-Drain control for this pin
13	GPIO45	R/W	0h	Output Open-Drain control for this pin

Table 7-88. GPBODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	GPIO44	R/W	0h	Output Open-Drain control for this pin
11	GPIO43	R/W	0h	Output Open-Drain control for this pin
10	GPIO42	R/W	0h	Output Open-Drain control for this pin
9	GPIO41	R/W	0h	Output Open-Drain control for this pin
8	GPIO40	R/W	0h	Output Open-Drain control for this pin
7	GPIO39	R/W	0h	Output Open-Drain control for this pin
6	GPIO38	R/W	0h	Output Open-Drain control for this pin
5	GPIO37	R/W	0h	Output Open-Drain control for this pin
4	GPIO36	R/W	0h	Output Open-Drain control for this pin
3	GPIO35	R/W	0h	Output Open-Drain control for this pin
2	GPIO34	R/W	0h	Output Open-Drain control for this pin
1	GPIO33	R/W	0h	Output Open-Drain control for this pin
0	GPIO32	R/W	0h	Output Open-Drain control for this pin

7.11.4.27 GPBAMSEL Register (Offset = 54h) [reset = 0h]

GPBAMSEL is shown in [Figure 7-78](#) and described in [Table 7-89](#).

GPIO B Analog Mode Select register

Selects between digital and analog functionality for GPIO pins.

0: The pin is configured to digital functions according to the other GPIO configuration registers

1: The analog function of the pin is enabled

Figure 7-78. GPBAMSEL Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	GPIO43	GPIO42	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R/W-0h	R/W-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-89. GPBAMSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	GPIO43	R/W	0h	Selects the USB0DP function
10	GPIO42	R/W	0h	Selects the USB0DM function
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved

Table 7-89. GPBAMSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

7.11.4.28 GPBGMUX1 Register (Offset = 60h) [reset = 0h]

GPBGMUX1 is shown in [Figure 7-79](#) and described in [Table 7-90](#).

GPIO B Peripheral Group Mux (GPIO32 to 47)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-79. 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		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-90. GPBGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO47	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO46	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO45	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO44	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO43	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO42	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO41	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO40	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO39	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO38	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO37	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO36	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO35	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO34	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO33	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO32	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.29 GPBGMUX2 Register (Offset = 62h) [reset = 0h]

GPBGMUX2 is shown in [Figure 7-80](#) and described in [Table 7-91](#).

GPIO B Peripheral Group Mux (GPIO48 to 63)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-80. GPBGMUX2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	
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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-91. GPBGMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO63	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO62	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO61	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO60	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO59	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO58	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO57	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO56	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO55	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO54	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO53	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO52	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO51	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO50	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO49	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO48	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.30 GPBCSEL1 Register (Offset = 68h) [reset = 0h]

GPBCSEL1 is shown in [Figure 7-81](#) and described in [Table 7-92](#).

GPIO B Core Select Register (GPIO32 to 39)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-81. GPBCSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO39				GPIO38				GPIO37				GPIO36			
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
GPIO35				GPIO34				GPIO33				GPIO32			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-92. GPBCSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO39	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO38	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO37	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO36	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO35	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO34	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO33	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO32	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.31 GPBCSEL2 Register (Offset = 6Ah) [reset = 0h]

GPBCSEL2 is shown in [Figure 7-82](#) and described in [Table 7-93](#).

GPIO B Core Select Register (GPIO40 to 47)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-82. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-93. GPBCSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO47	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO46	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO45	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO44	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO43	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO42	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO41	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO40	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.32 GPBCSEL3 Register (Offset = 6Ch) [reset = 0h]

GPBCSEL3 is shown in [Figure 7-83](#) and described in [Table 7-94](#).

GPIO B Core Select Register (GPIO48 to 55)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-83. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-94. GPBCSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO55	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO54	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO53	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO52	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO51	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO50	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO49	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO48	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.33 GPBCSEL4 Register (Offset = 6Eh) [reset = 0h]

GPBCSEL4 is shown in [Figure 7-84](#) and described in [Table 7-95](#).

GPIO B Core Select Register (GPIO56 to 63)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-84. GPBCSEL4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO63				GPIO62				GPIO61				GPIO60			
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
GPIO59				GPIO58				GPIO57				GPIO56			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-95. GPBCSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO63	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO62	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO61	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO60	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO59	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO58	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO57	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO56	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.34 GPBLOCK Register (Offset = 7Ch) [reset = 0h]

GPBLOCK is shown in [Figure 7-85](#) and described in [Table 7-96](#).

GPIO B Lock Configuration Register (GPIO32 to 63)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-85. GPBLOCK 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-96. GPBLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Configuration Lock bit for this pin
30	GPIO62	R/W	0h	Configuration Lock bit for this pin
29	GPIO61	R/W	0h	Configuration Lock bit for this pin
28	GPIO60	R/W	0h	Configuration Lock bit for this pin
27	GPIO59	R/W	0h	Configuration Lock bit for this pin
26	GPIO58	R/W	0h	Configuration Lock bit for this pin
25	GPIO57	R/W	0h	Configuration Lock bit for this pin
24	GPIO56	R/W	0h	Configuration Lock bit for this pin
23	GPIO55	R/W	0h	Configuration Lock bit for this pin
22	GPIO54	R/W	0h	Configuration Lock bit for this pin
21	GPIO53	R/W	0h	Configuration Lock bit for this pin
20	GPIO52	R/W	0h	Configuration Lock bit for this pin
19	GPIO51	R/W	0h	Configuration Lock bit for this pin
18	GPIO50	R/W	0h	Configuration Lock bit for this pin
17	GPIO49	R/W	0h	Configuration Lock bit for this pin
16	GPIO48	R/W	0h	Configuration Lock bit for this pin
15	GPIO47	R/W	0h	Configuration Lock bit for this pin
14	GPIO46	R/W	0h	Configuration Lock bit for this pin
13	GPIO45	R/W	0h	Configuration Lock bit for this pin
12	GPIO44	R/W	0h	Configuration Lock bit for this pin

Table 7-96. GPBLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	GPIO43	R/W	0h	Configuration Lock bit for this pin
10	GPIO42	R/W	0h	Configuration Lock bit for this pin
9	GPIO41	R/W	0h	Configuration Lock bit for this pin
8	GPIO40	R/W	0h	Configuration Lock bit for this pin
7	GPIO39	R/W	0h	Configuration Lock bit for this pin
6	GPIO38	R/W	0h	Configuration Lock bit for this pin
5	GPIO37	R/W	0h	Configuration Lock bit for this pin
4	GPIO36	R/W	0h	Configuration Lock bit for this pin
3	GPIO35	R/W	0h	Configuration Lock bit for this pin
2	GPIO34	R/W	0h	Configuration Lock bit for this pin
1	GPIO33	R/W	0h	Configuration Lock bit for this pin
0	GPIO32	R/W	0h	Configuration Lock bit for this pin

7.11.4.35 GPBCR Register (Offset = 7Eh) [reset = 0h]

GPBCR is shown in [Figure 7-86](#) and described in [Table 7-97](#).

GPIO B Lock Commit Register (GPIO32 to 63)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-86. GPBCR Register

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/WOnce-0h	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
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
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	GPIO38	GPIO37	GPIO36	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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-97. GPBCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/WOnce	0h	Configuration lock commit bit for this pin
30	GPIO62	R/WOnce	0h	Configuration lock commit bit for this pin
29	GPIO61	R/WOnce	0h	Configuration lock commit bit for this pin
28	GPIO60	R/WOnce	0h	Configuration lock commit bit for this pin
27	GPIO59	R/WOnce	0h	Configuration lock commit bit for this pin
26	GPIO58	R/WOnce	0h	Configuration lock commit bit for this pin
25	GPIO57	R/WOnce	0h	Configuration lock commit bit for this pin
24	GPIO56	R/WOnce	0h	Configuration lock commit bit for this pin
23	GPIO55	R/WOnce	0h	Configuration lock commit bit for this pin
22	GPIO54	R/WOnce	0h	Configuration lock commit bit for this pin
21	GPIO53	R/WOnce	0h	Configuration lock commit bit for this pin
20	GPIO52	R/WOnce	0h	Configuration lock commit bit for this pin
19	GPIO51	R/WOnce	0h	Configuration lock commit bit for this pin
18	GPIO50	R/WOnce	0h	Configuration lock commit bit for this pin
17	GPIO49	R/WOnce	0h	Configuration lock commit bit for this pin
16	GPIO48	R/WOnce	0h	Configuration lock commit bit for this pin
15	GPIO47	R/WOnce	0h	Configuration lock commit bit for this pin
14	GPIO46	R/WOnce	0h	Configuration lock commit bit for this pin
13	GPIO45	R/WOnce	0h	Configuration lock commit bit for this pin
12	GPIO44	R/WOnce	0h	Configuration lock commit bit for this pin
11	GPIO43	R/WOnce	0h	Configuration lock commit bit for this pin
10	GPIO42	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-97. GPBCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO41	R/WOnce	0h	Configuration lock commit bit for this pin
8	GPIO40	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO39	R/WOnce	0h	Configuration lock commit bit for this pin
6	GPIO38	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO37	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO36	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO35	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO34	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO33	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO32	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.4.36 GPCCTRL Register (Offset = 80h) [reset = 0h]

GPCCTRL is shown in [Figure 7-87](#) and described in [Table 7-98](#).

GPIO C Qualification Sampling Period Control (GPIO64 to 95)

Figure 7-87. GPCCTRL Register

31	30	29	28	27	26	25	24	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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-98. GPCCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO88 to GPIO95: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO80 to GPIO87: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO72 to GPIO79: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO64 to GPIO71: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510

7.11.4.37 GPCQSEL1 Register (Offset = 82h) [reset = 0h]

GPCQSEL1 is shown in [Figure 7-88](#) and described in [Table 7-99](#).

GPIO C Qualifier Select 1 Register (GPIO64 to 79)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-88. GPCQSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO79		GPIO78		GPIO77		GPIO76		GPIO75		GPIO74		GPIO73		GPIO72	
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
GPIO71		GPIO70		GPIO69		GPIO68		GPIO67		GPIO66		GPIO65		GPIO64	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-99. GPCQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO79	R/W	0h	Input qualification type
29-28	GPIO78	R/W	0h	Input qualification type
27-26	GPIO77	R/W	0h	Input qualification type
25-24	GPIO76	R/W	0h	Input qualification type
23-22	GPIO75	R/W	0h	Input qualification type
21-20	GPIO74	R/W	0h	Input qualification type
19-18	GPIO73	R/W	0h	Input qualification type
17-16	GPIO72	R/W	0h	Input qualification type
15-14	GPIO71	R/W	0h	Input qualification type
13-12	GPIO70	R/W	0h	Input qualification type
11-10	GPIO69	R/W	0h	Input qualification type
9-8	GPIO68	R/W	0h	Input qualification type
7-6	GPIO67	R/W	0h	Input qualification type
5-4	GPIO66	R/W	0h	Input qualification type
3-2	GPIO65	R/W	0h	Input qualification type
1-0	GPIO64	R/W	0h	Input qualification type

7.11.4.38 GPCQSEL2 Register (Offset = 84h) [reset = 0h]

GPCQSEL2 is shown in [Figure 7-89](#) and described in [Table 7-100](#).

GPIO C Qualifier Select 2 Register (GPIO80 to 95)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-89. GPCQSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO95		GPIO94		GPIO93		GPIO92		GPIO91		GPIO90		GPIO89		GPIO88	
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
GPIO87		GPIO86		GPIO85		GPIO84		GPIO83		GPIO82		GPIO81		GPIO80	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-100. GPCQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO95	R/W	0h	Input qualification type
29-28	GPIO94	R/W	0h	Input qualification type
27-26	GPIO93	R/W	0h	Input qualification type
25-24	GPIO92	R/W	0h	Input qualification type
23-22	GPIO91	R/W	0h	Input qualification type
21-20	GPIO90	R/W	0h	Input qualification type
19-18	GPIO89	R/W	0h	Input qualification type
17-16	GPIO88	R/W	0h	Input qualification type
15-14	GPIO87	R/W	0h	Input qualification type
13-12	GPIO86	R/W	0h	Input qualification type
11-10	GPIO85	R/W	0h	Input qualification type
9-8	GPIO84	R/W	0h	Input qualification type
7-6	GPIO83	R/W	0h	Input qualification type
5-4	GPIO82	R/W	0h	Input qualification type
3-2	GPIO81	R/W	0h	Input qualification type
1-0	GPIO80	R/W	0h	Input qualification type

7.11.4.39 GPCMUX1 Register (Offset = 86h) [reset = 0h]

GPCMUX1 is shown in [Figure 7-90](#) and described in [Table 7-101](#).

GPIO C Mux 1 Register (GPIO64 to 79)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-90. GPCMUX1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO79		GPIO78		GPIO77		GPIO76		GPIO75		GPIO74		GPIO73		GPIO72	
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
GPIO71		GPIO70		GPIO69		GPIO68		GPIO67		GPIO66		GPIO65		GPIO64	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-101. GPCMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO79	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO78	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO77	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO76	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO75	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO74	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO73	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO72	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO71	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO70	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO69	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO68	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO67	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO66	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO65	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO64	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.40 GPCMUX2 Register (Offset = 88h) [reset = 0h]

GPCMUX2 is shown in [Figure 7-91](#) and described in [Table 7-102](#).

GPIO C Mux 2 Register (GPIO80 to 95)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-91. GPCMUX2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO95		GPIO94		GPIO93		GPIO92		GPIO91		GPIO90		GPIO89		GPIO88	
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
GPIO87		GPIO86		GPIO85		GPIO84		GPIO83		GPIO82		GPIO81		GPIO80	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-102. GPCMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO95	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO94	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO93	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO92	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO91	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO90	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO89	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO88	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO87	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO86	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO85	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO84	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO83	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO82	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO81	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO80	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.41 GPCDIR Register (Offset = 8Ah) [reset = 0h]

GPCDIR is shown in [Figure 7-92](#) and described in [Table 7-103](#).

GPIO C Direction Register (GPIO64 to 95)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-92. GPCDIR Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-103. GPCDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Defines direction for this pin in GPIO mode
30	GPIO94	R/W	0h	Defines direction for this pin in GPIO mode
29	GPIO93	R/W	0h	Defines direction for this pin in GPIO mode
28	GPIO92	R/W	0h	Defines direction for this pin in GPIO mode
27	GPIO91	R/W	0h	Defines direction for this pin in GPIO mode
26	GPIO90	R/W	0h	Defines direction for this pin in GPIO mode
25	GPIO89	R/W	0h	Defines direction for this pin in GPIO mode
24	GPIO88	R/W	0h	Defines direction for this pin in GPIO mode
23	GPIO87	R/W	0h	Defines direction for this pin in GPIO mode
22	GPIO86	R/W	0h	Defines direction for this pin in GPIO mode
21	GPIO85	R/W	0h	Defines direction for this pin in GPIO mode
20	GPIO84	R/W	0h	Defines direction for this pin in GPIO mode
19	GPIO83	R/W	0h	Defines direction for this pin in GPIO mode
18	GPIO82	R/W	0h	Defines direction for this pin in GPIO mode
17	GPIO81	R/W	0h	Defines direction for this pin in GPIO mode
16	GPIO80	R/W	0h	Defines direction for this pin in GPIO mode
15	GPIO79	R/W	0h	Defines direction for this pin in GPIO mode
14	GPIO78	R/W	0h	Defines direction for this pin in GPIO mode
13	GPIO77	R/W	0h	Defines direction for this pin in GPIO mode
12	GPIO76	R/W	0h	Defines direction for this pin in GPIO mode
11	GPIO75	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-103. GPCDIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO74	R/W	0h	Defines direction for this pin in GPIO mode
9	GPIO73	R/W	0h	Defines direction for this pin in GPIO mode
8	GPIO72	R/W	0h	Defines direction for this pin in GPIO mode
7	GPIO71	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO70	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO69	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO68	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO67	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO66	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO65	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO64	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.42 GPCPUD Register (Offset = 8Ch) [reset = FFFFFFFFh]

GPCPUD is shown in [Figure 7-93](#) and described in [Table 7-104](#).

GPIO C Pull Up Disable Register (GPIO64 to 95)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-93. GPCPUD Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-104. GPCPUD Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	1h	Pull-Up Disable control for this pin
30	GPIO94	R/W	1h	Pull-Up Disable control for this pin
29	GPIO93	R/W	1h	Pull-Up Disable control for this pin
28	GPIO92	R/W	1h	Pull-Up Disable control for this pin
27	GPIO91	R/W	1h	Pull-Up Disable control for this pin
26	GPIO90	R/W	1h	Pull-Up Disable control for this pin
25	GPIO89	R/W	1h	Pull-Up Disable control for this pin
24	GPIO88	R/W	1h	Pull-Up Disable control for this pin
23	GPIO87	R/W	1h	Pull-Up Disable control for this pin
22	GPIO86	R/W	1h	Pull-Up Disable control for this pin
21	GPIO85	R/W	1h	Pull-Up Disable control for this pin
20	GPIO84	R/W	1h	Pull-Up Disable control for this pin
19	GPIO83	R/W	1h	Pull-Up Disable control for this pin
18	GPIO82	R/W	1h	Pull-Up Disable control for this pin
17	GPIO81	R/W	1h	Pull-Up Disable control for this pin
16	GPIO80	R/W	1h	Pull-Up Disable control for this pin
15	GPIO79	R/W	1h	Pull-Up Disable control for this pin
14	GPIO78	R/W	1h	Pull-Up Disable control for this pin
13	GPIO77	R/W	1h	Pull-Up Disable control for this pin
12	GPIO76	R/W	1h	Pull-Up Disable control for this pin
11	GPIO75	R/W	1h	Pull-Up Disable control for this pin

Table 7-104. GPCPUD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO74	R/W	1h	Pull-Up Disable control for this pin
9	GPIO73	R/W	1h	Pull-Up Disable control for this pin
8	GPIO72	R/W	1h	Pull-Up Disable control for this pin
7	GPIO71	R/W	1h	Pull-Up Disable control for this pin
6	GPIO70	R/W	1h	Pull-Up Disable control for this pin
5	GPIO69	R/W	1h	Pull-Up Disable control for this pin
4	GPIO68	R/W	1h	Pull-Up Disable control for this pin
3	GPIO67	R/W	1h	Pull-Up Disable control for this pin
2	GPIO66	R/W	1h	Pull-Up Disable control for this pin
1	GPIO65	R/W	1h	Pull-Up Disable control for this pin
0	GPIO64	R/W	1h	Pull-Up Disable control for this pin

7.11.4.43 GPCINV Register (Offset = 90h) [reset = 0h]

GPCINV is shown in [Figure 7-94](#) and described in [Table 7-105](#).

GPIO C Input Polarity Invert Registers (GPIO64 to 95)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-94. GPCINV Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-105. GPCINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Input inversion control for this pin
30	GPIO94	R/W	0h	Input inversion control for this pin
29	GPIO93	R/W	0h	Input inversion control for this pin
28	GPIO92	R/W	0h	Input inversion control for this pin
27	GPIO91	R/W	0h	Input inversion control for this pin
26	GPIO90	R/W	0h	Input inversion control for this pin
25	GPIO89	R/W	0h	Input inversion control for this pin
24	GPIO88	R/W	0h	Input inversion control for this pin
23	GPIO87	R/W	0h	Input inversion control for this pin
22	GPIO86	R/W	0h	Input inversion control for this pin
21	GPIO85	R/W	0h	Input inversion control for this pin
20	GPIO84	R/W	0h	Input inversion control for this pin
19	GPIO83	R/W	0h	Input inversion control for this pin
18	GPIO82	R/W	0h	Input inversion control for this pin
17	GPIO81	R/W	0h	Input inversion control for this pin
16	GPIO80	R/W	0h	Input inversion control for this pin
15	GPIO79	R/W	0h	Input inversion control for this pin
14	GPIO78	R/W	0h	Input inversion control for this pin
13	GPIO77	R/W	0h	Input inversion control for this pin
12	GPIO76	R/W	0h	Input inversion control for this pin
11	GPIO75	R/W	0h	Input inversion control for this pin

Table 7-105. GPCINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO74	R/W	0h	Input inversion control for this pin
9	GPIO73	R/W	0h	Input inversion control for this pin
8	GPIO72	R/W	0h	Input inversion control for this pin
7	GPIO71	R/W	0h	Input inversion control for this pin
6	GPIO70	R/W	0h	Input inversion control for this pin
5	GPIO69	R/W	0h	Input inversion control for this pin
4	GPIO68	R/W	0h	Input inversion control for this pin
3	GPIO67	R/W	0h	Input inversion control for this pin
2	GPIO66	R/W	0h	Input inversion control for this pin
1	GPIO65	R/W	0h	Input inversion control for this pin
0	GPIO64	R/W	0h	Input inversion control for this pin

7.11.4.44 GPCODR Register (Offset = 92h) [reset = 0h]

GPCODR is shown in [Figure 7-95](#) and described in [Table 7-106](#).

GPIO C Open Drain Output Register (GPIO64 to GPIO95)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-95. GPCODR Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-106. GPCODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Output Open-Drain control for this pin
30	GPIO94	R/W	0h	Output Open-Drain control for this pin
29	GPIO93	R/W	0h	Output Open-Drain control for this pin
28	GPIO92	R/W	0h	Output Open-Drain control for this pin
27	GPIO91	R/W	0h	Output Open-Drain control for this pin
26	GPIO90	R/W	0h	Output Open-Drain control for this pin
25	GPIO89	R/W	0h	Output Open-Drain control for this pin
24	GPIO88	R/W	0h	Output Open-Drain control for this pin
23	GPIO87	R/W	0h	Output Open-Drain control for this pin
22	GPIO86	R/W	0h	Output Open-Drain control for this pin
21	GPIO85	R/W	0h	Output Open-Drain control for this pin
20	GPIO84	R/W	0h	Output Open-Drain control for this pin
19	GPIO83	R/W	0h	Output Open-Drain control for this pin
18	GPIO82	R/W	0h	Output Open-Drain control for this pin
17	GPIO81	R/W	0h	Output Open-Drain control for this pin
16	GPIO80	R/W	0h	Output Open-Drain control for this pin
15	GPIO79	R/W	0h	Output Open-Drain control for this pin
14	GPIO78	R/W	0h	Output Open-Drain control for this pin
13	GPIO77	R/W	0h	Output Open-Drain control for this pin

Table 7-106. GPCODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	GPIO76	R/W	0h	Output Open-Drain control for this pin
11	GPIO75	R/W	0h	Output Open-Drain control for this pin
10	GPIO74	R/W	0h	Output Open-Drain control for this pin
9	GPIO73	R/W	0h	Output Open-Drain control for this pin
8	GPIO72	R/W	0h	Output Open-Drain control for this pin
7	GPIO71	R/W	0h	Output Open-Drain control for this pin
6	GPIO70	R/W	0h	Output Open-Drain control for this pin
5	GPIO69	R/W	0h	Output Open-Drain control for this pin
4	GPIO68	R/W	0h	Output Open-Drain control for this pin
3	GPIO67	R/W	0h	Output Open-Drain control for this pin
2	GPIO66	R/W	0h	Output Open-Drain control for this pin
1	GPIO65	R/W	0h	Output Open-Drain control for this pin
0	GPIO64	R/W	0h	Output Open-Drain control for this pin

7.11.4.45 GPCGMUX1 Register (Offset = A0h) [reset = 0h]

GPCGMUX1 is shown in [Figure 7-96](#) and described in [Table 7-107](#).

GPIO C Peripheral Group Mux (GPIO64 to 79)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-96. GPCGMUX1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO79		GPIO78		GPIO77		GPIO76		GPIO75		GPIO74		GPIO73		GPIO72	
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
GPIO71		GPIO70		GPIO69		GPIO68		GPIO67		GPIO66		GPIO65		GPIO64	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-107. GPCGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO79	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO78	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO77	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO76	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO75	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO74	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO73	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO72	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO71	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO70	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO69	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO68	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO67	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO66	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO65	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO64	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.46 GPCGMUX2 Register (Offset = A2h) [reset = 0h]

GPCGMUX2 is shown in [Figure 7-97](#) and described in [Table 7-108](#).

GPIO C Peripheral Group Mux (GPIO80 to 95)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-97. GPCGMUX2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88								
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80								
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h								

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-108. GPCGMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO95	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO94	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO93	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO92	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO91	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO90	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO89	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO88	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO87	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO86	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO85	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO84	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO83	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO82	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO81	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO80	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.47 GPCCSSEL1 Register (Offset = A8h) [reset = 0h]

GPCCSSEL1 is shown in [Figure 7-98](#) and described in [Table 7-109](#).

GPIO C Core Select Register (GPIO64 to 71)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-98. GPCCSSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO71				GPIO70				GPIO69				GPIO68			
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
GPIO67				GPIO66				GPIO65				GPIO64			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-109. GPCCSSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO71	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO70	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO69	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO68	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO67	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO66	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO65	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO64	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.48 GPCCSSEL2 Register (Offset = AAh) [reset = 0h]

GPCCSSEL2 is shown in [Figure 7-99](#) and described in [Table 7-110](#).

GPIO C Core Select Register (GPIO72 to 79)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-99. GPCCSSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO79				GPIO78				GPIO77				GPIO76			
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
GPIO75				GPIO74				GPIO73				GPIO72			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-110. GPCCSSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO79	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO78	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO77	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO76	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO75	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO74	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO73	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO72	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.49 GPCCSSEL3 Register (Offset = ACh) [reset = 0h]

GPCCSSEL3 is shown in [Figure 7-100](#) and described in [Table 7-111](#).

GPIO C Core Select Register (GPIO80 to 87)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-100. GPCCSSEL3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO87				GPIO86				GPIO85				GPIO84			
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
GPIO83				GPIO82				GPIO81				GPIO80			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-111. GPCCSSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO87	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO86	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO85	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO84	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO83	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO82	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO81	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO80	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.50 GPCCCSEL4 Register (Offset = AEh) [reset = 0h]

GPCCCSEL4 is shown in [Figure 7-101](#) and described in [Table 7-112](#).

GPIO C Core Select Register (GPIO88 to 95)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-101. GPCCCSEL4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO95				GPIO94				GPIO93				GPIO92			
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
GPIO91				GPIO90				GPIO89				GPIO88			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-112. GPCCCSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO95	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO94	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO93	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO92	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO91	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO90	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO89	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO88	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.51 GPCLOCK Register (Offset = BCh) [reset = 0h]

GPCLOCK is shown in [Figure 7-102](#) and described in [Table 7-113](#).

GPIO C Lock Configuration Register (GPIO64 to 95)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-102. GPCLOCK Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-113. GPCLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Configuration Lock bit for this pin
30	GPIO94	R/W	0h	Configuration Lock bit for this pin
29	GPIO93	R/W	0h	Configuration Lock bit for this pin
28	GPIO92	R/W	0h	Configuration Lock bit for this pin
27	GPIO91	R/W	0h	Configuration Lock bit for this pin
26	GPIO90	R/W	0h	Configuration Lock bit for this pin
25	GPIO89	R/W	0h	Configuration Lock bit for this pin
24	GPIO88	R/W	0h	Configuration Lock bit for this pin
23	GPIO87	R/W	0h	Configuration Lock bit for this pin
22	GPIO86	R/W	0h	Configuration Lock bit for this pin
21	GPIO85	R/W	0h	Configuration Lock bit for this pin
20	GPIO84	R/W	0h	Configuration Lock bit for this pin
19	GPIO83	R/W	0h	Configuration Lock bit for this pin
18	GPIO82	R/W	0h	Configuration Lock bit for this pin
17	GPIO81	R/W	0h	Configuration Lock bit for this pin
16	GPIO80	R/W	0h	Configuration Lock bit for this pin
15	GPIO79	R/W	0h	Configuration Lock bit for this pin
14	GPIO78	R/W	0h	Configuration Lock bit for this pin
13	GPIO77	R/W	0h	Configuration Lock bit for this pin
12	GPIO76	R/W	0h	Configuration Lock bit for this pin

Table 7-113. GPCLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	GPIO75	R/W	0h	Configuration Lock bit for this pin
10	GPIO74	R/W	0h	Configuration Lock bit for this pin
9	GPIO73	R/W	0h	Configuration Lock bit for this pin
8	GPIO72	R/W	0h	Configuration Lock bit for this pin
7	GPIO71	R/W	0h	Configuration Lock bit for this pin
6	GPIO70	R/W	0h	Configuration Lock bit for this pin
5	GPIO69	R/W	0h	Configuration Lock bit for this pin
4	GPIO68	R/W	0h	Configuration Lock bit for this pin
3	GPIO67	R/W	0h	Configuration Lock bit for this pin
2	GPIO66	R/W	0h	Configuration Lock bit for this pin
1	GPIO65	R/W	0h	Configuration Lock bit for this pin
0	GPIO64	R/W	0h	Configuration Lock bit for this pin

7.11.4.52 GPCCR Register (Offset = BEh) [reset = 0h]

GPCCR is shown in [Figure 7-103](#) and described in [Table 7-114](#).

GPIO C Lock Commit Register (GPIO64 to 95)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-103. GPCCR Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
R/WOnce-0h	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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-114. GPCCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/WOnce	0h	Configuration lock commit bit for this pin
30	GPIO94	R/WOnce	0h	Configuration lock commit bit for this pin
29	GPIO93	R/WOnce	0h	Configuration lock commit bit for this pin
28	GPIO92	R/WOnce	0h	Configuration lock commit bit for this pin
27	GPIO91	R/WOnce	0h	Configuration lock commit bit for this pin
26	GPIO90	R/WOnce	0h	Configuration lock commit bit for this pin
25	GPIO89	R/WOnce	0h	Configuration lock commit bit for this pin
24	GPIO88	R/WOnce	0h	Configuration lock commit bit for this pin
23	GPIO87	R/WOnce	0h	Configuration lock commit bit for this pin
22	GPIO86	R/WOnce	0h	Configuration lock commit bit for this pin
21	GPIO85	R/WOnce	0h	Configuration lock commit bit for this pin
20	GPIO84	R/WOnce	0h	Configuration lock commit bit for this pin
19	GPIO83	R/WOnce	0h	Configuration lock commit bit for this pin
18	GPIO82	R/WOnce	0h	Configuration lock commit bit for this pin
17	GPIO81	R/WOnce	0h	Configuration lock commit bit for this pin
16	GPIO80	R/WOnce	0h	Configuration lock commit bit for this pin
15	GPIO79	R/WOnce	0h	Configuration lock commit bit for this pin
14	GPIO78	R/WOnce	0h	Configuration lock commit bit for this pin
13	GPIO77	R/WOnce	0h	Configuration lock commit bit for this pin
12	GPIO76	R/WOnce	0h	Configuration lock commit bit for this pin
11	GPIO75	R/WOnce	0h	Configuration lock commit bit for this pin
10	GPIO74	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-114. GPCCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO73	R/WOnce	0h	Configuration lock commit bit for this pin
8	GPIO72	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO71	R/WOnce	0h	Configuration lock commit bit for this pin
6	GPIO70	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO69	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO68	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO67	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO66	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO65	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO64	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.4.53 GPDCTRL Register (Offset = C0h) [reset = 0h]

GPDCTRL is shown in [Figure 7-104](#) and described in [Table 7-115](#).

GPIO D Qualification Sampling Period Control (GPIO96 to 127)

Figure 7-104. GPDCTRL Register

31	30	29	28	27	26	25	24	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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-115. GPDCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO120 to GPIO127: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO112 to GPIO119: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO104 to GPIO111: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO96 to GPIO103: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510

7.11.4.54 GPDQSEL1 Register (Offset = C2h) [reset = 0h]

GPDQSEL1 is shown in [Figure 7-105](#) and described in [Table 7-116](#).

GPIO D Qualifier Select 1 Register (GPIO96 to 111)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-105. GPDQSEL1 Register

31	30	29	28	27	26	25	24
GPIO111		GPIO110		GPIO109		GPIO108	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO107		GPIO106		GPIO105		GPIO104	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO103		GPIO102		GPIO101		GPIO100	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO99		GPIO98		GPIO97		GPIO96	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-116. GPDQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO111	R/W	0h	Input qualification type
29-28	GPIO110	R/W	0h	Input qualification type
27-26	GPIO109	R/W	0h	Input qualification type
25-24	GPIO108	R/W	0h	Input qualification type
23-22	GPIO107	R/W	0h	Input qualification type
21-20	GPIO106	R/W	0h	Input qualification type
19-18	GPIO105	R/W	0h	Input qualification type
17-16	GPIO104	R/W	0h	Input qualification type
15-14	GPIO103	R/W	0h	Input qualification type
13-12	GPIO102	R/W	0h	Input qualification type
11-10	GPIO101	R/W	0h	Input qualification type
9-8	GPIO100	R/W	0h	Input qualification type
7-6	GPIO99	R/W	0h	Input qualification type
5-4	GPIO98	R/W	0h	Input qualification type
3-2	GPIO97	R/W	0h	Input qualification type
1-0	GPIO96	R/W	0h	Input qualification type

7.11.4.55 GPDQSEL2 Register (Offset = C4h) [reset = 0h]

GPDQSEL2 is shown in [Figure 7-106](#) and described in [Table 7-117](#).

GPIO D Qualifier Select 2 Register (GPIO112 to 127)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-106. GPDQSEL2 Register

31	30	29	28	27	26	25	24
GPIO127		GPIO126		GPIO125		GPIO124	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO123		GPIO122		GPIO121		GPIO120	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO119		GPIO118		GPIO117		GPIO116	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO115		GPIO114		GPIO113		GPIO112	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-117. GPDQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO127	R/W	0h	Input qualification type
29-28	GPIO126	R/W	0h	Input qualification type
27-26	GPIO125	R/W	0h	Input qualification type
25-24	GPIO124	R/W	0h	Input qualification type
23-22	GPIO123	R/W	0h	Input qualification type
21-20	GPIO122	R/W	0h	Input qualification type
19-18	GPIO121	R/W	0h	Input qualification type
17-16	GPIO120	R/W	0h	Input qualification type
15-14	GPIO119	R/W	0h	Input qualification type
13-12	GPIO118	R/W	0h	Input qualification type
11-10	GPIO117	R/W	0h	Input qualification type
9-8	GPIO116	R/W	0h	Input qualification type
7-6	GPIO115	R/W	0h	Input qualification type
5-4	GPIO114	R/W	0h	Input qualification type
3-2	GPIO113	R/W	0h	Input qualification type
1-0	GPIO112	R/W	0h	Input qualification type

7.11.4.56 GPDMUX1 Register (Offset = C6h) [reset = 0h]

GPDMUX1 is shown in [Figure 7-107](#) and described in [Table 7-118](#).

GPIO D Mux 1 Register (GPIO96 to 111)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-107. GPDMUX1 Register

31	30	29	28	27	26	25	24
GPIO111		GPIO110		GPIO109		GPIO108	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO107		GPIO106		GPIO105		GPIO104	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO103		GPIO102		GPIO101		GPIO100	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO99		GPIO98		GPIO97		GPIO96	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-118. GPDMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO111	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO110	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO109	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO108	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO107	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO106	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO105	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO104	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO103	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO102	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO101	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO100	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO99	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO98	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO97	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO96	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.57 GPDMUX2 Register (Offset = C8h) [reset = 0h]

GPDMUX2 is shown in [Figure 7-108](#) and described in [Table 7-119](#).

GPIO D Mux 2 Register (GPIO112 to 127)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-108. GPDMUX2 Register

31	30	29	28	27	26	25	24
GPIO127		GPIO126		GPIO125		GPIO124	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO123		GPIO122		GPIO121		GPIO120	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO119		GPIO118		GPIO117		GPIO116	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO115		GPIO114		GPIO113		GPIO112	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-119. GPDMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO127	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO126	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO125	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO124	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO123	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO122	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO121	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO120	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO119	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO118	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO117	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO116	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO115	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO114	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO113	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO112	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.58 GPDDIR Register (Offset = CAh) [reset = 0h]

GPDDIR is shown in [Figure 7-109](#) and described in [Table 7-120](#).

GPIO D Direction Register (GPIO96 to 127)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-109. GPDDIR Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-120. GPDDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Defines direction for this pin in GPIO mode
30	GPIO126	R/W	0h	Defines direction for this pin in GPIO mode
29	GPIO125	R/W	0h	Defines direction for this pin in GPIO mode
28	GPIO124	R/W	0h	Defines direction for this pin in GPIO mode
27	GPIO123	R/W	0h	Defines direction for this pin in GPIO mode
26	GPIO122	R/W	0h	Defines direction for this pin in GPIO mode
25	GPIO121	R/W	0h	Defines direction for this pin in GPIO mode
24	GPIO120	R/W	0h	Defines direction for this pin in GPIO mode
23	GPIO119	R/W	0h	Defines direction for this pin in GPIO mode
22	GPIO118	R/W	0h	Defines direction for this pin in GPIO mode
21	GPIO117	R/W	0h	Defines direction for this pin in GPIO mode
20	GPIO116	R/W	0h	Defines direction for this pin in GPIO mode
19	GPIO115	R/W	0h	Defines direction for this pin in GPIO mode
18	GPIO114	R/W	0h	Defines direction for this pin in GPIO mode
17	GPIO113	R/W	0h	Defines direction for this pin in GPIO mode
16	GPIO112	R/W	0h	Defines direction for this pin in GPIO mode
15	GPIO111	R/W	0h	Defines direction for this pin in GPIO mode
14	GPIO110	R/W	0h	Defines direction for this pin in GPIO mode
13	GPIO109	R/W	0h	Defines direction for this pin in GPIO mode
12	GPIO108	R/W	0h	Defines direction for this pin in GPIO mode
11	GPIO107	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-120. GPDDIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO106	R/W	0h	Defines direction for this pin in GPIO mode
9	GPIO105	R/W	0h	Defines direction for this pin in GPIO mode
8	GPIO104	R/W	0h	Defines direction for this pin in GPIO mode
7	GPIO103	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO102	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO101	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO100	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO99	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO98	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO97	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO96	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.59 GPDPU Register (Offset = CCh) [reset = FFFFFFFFh]

GPDPU is shown in [Figure 7-110](#) and described in [Table 7-121](#).

GPIO D Pull Up Disable Register (GPIO96 to 127)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-110. GPDPU Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-121. GPDPU Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	1h	Pull-Up Disable control for this pin
30	GPIO126	R/W	1h	Pull-Up Disable control for this pin
29	GPIO125	R/W	1h	Pull-Up Disable control for this pin
28	GPIO124	R/W	1h	Pull-Up Disable control for this pin
27	GPIO123	R/W	1h	Pull-Up Disable control for this pin
26	GPIO122	R/W	1h	Pull-Up Disable control for this pin
25	GPIO121	R/W	1h	Pull-Up Disable control for this pin
24	GPIO120	R/W	1h	Pull-Up Disable control for this pin
23	GPIO119	R/W	1h	Pull-Up Disable control for this pin
22	GPIO118	R/W	1h	Pull-Up Disable control for this pin
21	GPIO117	R/W	1h	Pull-Up Disable control for this pin
20	GPIO116	R/W	1h	Pull-Up Disable control for this pin
19	GPIO115	R/W	1h	Pull-Up Disable control for this pin
18	GPIO114	R/W	1h	Pull-Up Disable control for this pin
17	GPIO113	R/W	1h	Pull-Up Disable control for this pin
16	GPIO112	R/W	1h	Pull-Up Disable control for this pin
15	GPIO111	R/W	1h	Pull-Up Disable control for this pin
14	GPIO110	R/W	1h	Pull-Up Disable control for this pin
13	GPIO109	R/W	1h	Pull-Up Disable control for this pin
12	GPIO108	R/W	1h	Pull-Up Disable control for this pin
11	GPIO107	R/W	1h	Pull-Up Disable control for this pin

Table 7-121. GPDPU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO106	R/W	1h	Pull-Up Disable control for this pin
9	GPIO105	R/W	1h	Pull-Up Disable control for this pin
8	GPIO104	R/W	1h	Pull-Up Disable control for this pin
7	GPIO103	R/W	1h	Pull-Up Disable control for this pin
6	GPIO102	R/W	1h	Pull-Up Disable control for this pin
5	GPIO101	R/W	1h	Pull-Up Disable control for this pin
4	GPIO100	R/W	1h	Pull-Up Disable control for this pin
3	GPIO99	R/W	1h	Pull-Up Disable control for this pin
2	GPIO98	R/W	1h	Pull-Up Disable control for this pin
1	GPIO97	R/W	1h	Pull-Up Disable control for this pin
0	GPIO96	R/W	1h	Pull-Up Disable control for this pin

7.11.4.60 GPDINV Register (Offset = D0h) [reset = 0h]

GPDINV is shown in [Figure 7-111](#) and described in [Table 7-122](#).

GPIO D Input Polarity Invert Registers (GPIO96 to 127)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-111. GPDINV Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-122. GPDINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Input inversion control for this pin
30	GPIO126	R/W	0h	Input inversion control for this pin
29	GPIO125	R/W	0h	Input inversion control for this pin
28	GPIO124	R/W	0h	Input inversion control for this pin
27	GPIO123	R/W	0h	Input inversion control for this pin
26	GPIO122	R/W	0h	Input inversion control for this pin
25	GPIO121	R/W	0h	Input inversion control for this pin
24	GPIO120	R/W	0h	Input inversion control for this pin
23	GPIO119	R/W	0h	Input inversion control for this pin
22	GPIO118	R/W	0h	Input inversion control for this pin
21	GPIO117	R/W	0h	Input inversion control for this pin
20	GPIO116	R/W	0h	Input inversion control for this pin
19	GPIO115	R/W	0h	Input inversion control for this pin
18	GPIO114	R/W	0h	Input inversion control for this pin
17	GPIO113	R/W	0h	Input inversion control for this pin
16	GPIO112	R/W	0h	Input inversion control for this pin
15	GPIO111	R/W	0h	Input inversion control for this pin
14	GPIO110	R/W	0h	Input inversion control for this pin
13	GPIO109	R/W	0h	Input inversion control for this pin
12	GPIO108	R/W	0h	Input inversion control for this pin
11	GPIO107	R/W	0h	Input inversion control for this pin

Table 7-122. GPDINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO106	R/W	0h	Input inversion control for this pin
9	GPIO105	R/W	0h	Input inversion control for this pin
8	GPIO104	R/W	0h	Input inversion control for this pin
7	GPIO103	R/W	0h	Input inversion control for this pin
6	GPIO102	R/W	0h	Input inversion control for this pin
5	GPIO101	R/W	0h	Input inversion control for this pin
4	GPIO100	R/W	0h	Input inversion control for this pin
3	GPIO99	R/W	0h	Input inversion control for this pin
2	GPIO98	R/W	0h	Input inversion control for this pin
1	GPIO97	R/W	0h	Input inversion control for this pin
0	GPIO96	R/W	0h	Input inversion control for this pin

7.11.4.61 GPDODR Register (Offset = D2h) [reset = 0h]

GPDODR is shown in [Figure 7-112](#) and described in [Table 7-123](#).

GPIO D Open Drain Output Register (GPIO96 to GPIO127)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-112. GPDODR Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-123. GPDODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Output Open-Drain control for this pin
30	GPIO126	R/W	0h	Output Open-Drain control for this pin
29	GPIO125	R/W	0h	Output Open-Drain control for this pin
28	GPIO124	R/W	0h	Output Open-Drain control for this pin
27	GPIO123	R/W	0h	Output Open-Drain control for this pin
26	GPIO122	R/W	0h	Output Open-Drain control for this pin
25	GPIO121	R/W	0h	Output Open-Drain control for this pin
24	GPIO120	R/W	0h	Output Open-Drain control for this pin
23	GPIO119	R/W	0h	Output Open-Drain control for this pin
22	GPIO118	R/W	0h	Output Open-Drain control for this pin
21	GPIO117	R/W	0h	Output Open-Drain control for this pin
20	GPIO116	R/W	0h	Output Open-Drain control for this pin
19	GPIO115	R/W	0h	Output Open-Drain control for this pin
18	GPIO114	R/W	0h	Output Open-Drain control for this pin
17	GPIO113	R/W	0h	Output Open-Drain control for this pin
16	GPIO112	R/W	0h	Output Open-Drain control for this pin
15	GPIO111	R/W	0h	Output Open-Drain control for this pin
14	GPIO110	R/W	0h	Output Open-Drain control for this pin
13	GPIO109	R/W	0h	Output Open-Drain control for this pin

Table 7-123. GPDODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	GPIO108	R/W	0h	Output Open-Drain control for this pin
11	GPIO107	R/W	0h	Output Open-Drain control for this pin
10	GPIO106	R/W	0h	Output Open-Drain control for this pin
9	GPIO105	R/W	0h	Output Open-Drain control for this pin
8	GPIO104	R/W	0h	Output Open-Drain control for this pin
7	GPIO103	R/W	0h	Output Open-Drain control for this pin
6	GPIO102	R/W	0h	Output Open-Drain control for this pin
5	GPIO101	R/W	0h	Output Open-Drain control for this pin
4	GPIO100	R/W	0h	Output Open-Drain control for this pin
3	GPIO99	R/W	0h	Output Open-Drain control for this pin
2	GPIO98	R/W	0h	Output Open-Drain control for this pin
1	GPIO97	R/W	0h	Output Open-Drain control for this pin
0	GPIO96	R/W	0h	Output Open-Drain control for this pin

7.11.4.62 GPDGMUX1 Register (Offset = E0h) [reset = 0h]

GPDGMUX1 is shown in [Figure 7-113](#) and described in [Table 7-124](#).

GPIO D Peripheral Group Mux (GPIO96 to 111)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-113. GPDGMUX1 Register

31	30	29	28	27	26	25	24
GPIO111		GPIO110		GPIO109		GPIO108	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO107		GPIO106		GPIO105		GPIO104	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO103		GPIO102		GPIO101		GPIO100	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO99		GPIO98		GPIO97		GPIO96	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-124. GPDGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO111	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO110	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO109	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO108	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO107	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO106	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO105	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO104	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO103	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO102	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO101	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO100	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO99	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO98	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO97	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO96	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.63 GPDGMUX2 Register (Offset = E2h) [reset = 0h]

GPDGMUX2 is shown in [Figure 7-114](#) and described in [Table 7-125](#).

GPIO D Peripheral Group Mux (GPIO112 to 127)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-114. GPDGMUX2 Register

31	30	29	28	27	26	25	24
GPIO127		GPIO126		GPIO125		GPIO124	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO123		GPIO122		GPIO121		GPIO120	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO119		GPIO118		GPIO117		GPIO116	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO115		GPIO114		GPIO113		GPIO112	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-125. GPDGMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO127	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO126	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO125	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO124	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO123	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO122	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO121	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO120	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO119	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO118	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO117	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO116	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO115	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO114	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO113	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO112	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.64 GPDCSEL1 Register (Offset = E8h) [reset = 0h]

GPDCSEL1 is shown in [Figure 7-115](#) and described in [Table 7-126](#).

GPIO D Core Select Register (GPIO96 to 103)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-115. GPDCSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO103				GPIO102				GPIO101				GPIO100			
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
GPIO99				GPIO98				GPIO97				GPIO96			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-126. GPDCSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO103	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO102	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO101	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO100	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO99	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO98	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO97	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO96	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.65 GPDCSEL2 Register (Offset = EAh) [reset = 0h]

GPDCSEL2 is shown in [Figure 7-116](#) and described in [Table 7-127](#).

GPIO D Core Select Register (GPIO104 to 111)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-116. GPDCSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO111				GPIO110				GPIO109				GPIO108			
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
GPIO107				GPIO106				GPIO105				GPIO104			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-127. GPDCSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO111	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO110	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO109	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO108	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO107	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO106	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO105	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO104	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.66 GPDCSEL3 Register (Offset = ECh) [reset = 0h]

GPDCSEL3 is shown in [Figure 7-117](#) and described in [Table 7-128](#).

GPIO D Core Select Register (GPIO112 to 119)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-117. GPDCSEL3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO119				GPIO118				GPIO117				GPIO116			
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
GPIO115				GPIO114				GPIO113				GPIO112			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-128. GPDCSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO119	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO118	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO117	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO116	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO115	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO114	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO113	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO112	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.67 GPDCEL4 Register (Offset = EEh) [reset = 0h]

GPDCEL4 is shown in [Figure 7-118](#) and described in [Table 7-129](#).

GPIO D Core Select Register (GPIO120 to 127)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-118. GPDCEL4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO127				GPIO126				GPIO125				GPIO124			
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
GPIO123				GPIO122				GPIO121				GPIO120			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-129. GPDCEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO127	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO126	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO125	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO124	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO123	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO122	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO121	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO120	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.68 GPDLOCK Register (Offset = FCh) [reset = 0h]

GPDLOCK is shown in [Figure 7-119](#) and described in [Table 7-130](#).

GPIO D Lock Configuration Register (GPIO96 to 127)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-119. GPDLOCK Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-130. GPDLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Configuration Lock bit for this pin
30	GPIO126	R/W	0h	Configuration Lock bit for this pin
29	GPIO125	R/W	0h	Configuration Lock bit for this pin
28	GPIO124	R/W	0h	Configuration Lock bit for this pin
27	GPIO123	R/W	0h	Configuration Lock bit for this pin
26	GPIO122	R/W	0h	Configuration Lock bit for this pin
25	GPIO121	R/W	0h	Configuration Lock bit for this pin
24	GPIO120	R/W	0h	Configuration Lock bit for this pin
23	GPIO119	R/W	0h	Configuration Lock bit for this pin
22	GPIO118	R/W	0h	Configuration Lock bit for this pin
21	GPIO117	R/W	0h	Configuration Lock bit for this pin
20	GPIO116	R/W	0h	Configuration Lock bit for this pin
19	GPIO115	R/W	0h	Configuration Lock bit for this pin
18	GPIO114	R/W	0h	Configuration Lock bit for this pin
17	GPIO113	R/W	0h	Configuration Lock bit for this pin
16	GPIO112	R/W	0h	Configuration Lock bit for this pin
15	GPIO111	R/W	0h	Configuration Lock bit for this pin
14	GPIO110	R/W	0h	Configuration Lock bit for this pin
13	GPIO109	R/W	0h	Configuration Lock bit for this pin
12	GPIO108	R/W	0h	Configuration Lock bit for this pin

Table 7-130. GPDLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	GPIO107	R/W	0h	Configuration Lock bit for this pin
10	GPIO106	R/W	0h	Configuration Lock bit for this pin
9	GPIO105	R/W	0h	Configuration Lock bit for this pin
8	GPIO104	R/W	0h	Configuration Lock bit for this pin
7	GPIO103	R/W	0h	Configuration Lock bit for this pin
6	GPIO102	R/W	0h	Configuration Lock bit for this pin
5	GPIO101	R/W	0h	Configuration Lock bit for this pin
4	GPIO100	R/W	0h	Configuration Lock bit for this pin
3	GPIO99	R/W	0h	Configuration Lock bit for this pin
2	GPIO98	R/W	0h	Configuration Lock bit for this pin
1	GPIO97	R/W	0h	Configuration Lock bit for this pin
0	GPIO96	R/W	0h	Configuration Lock bit for this pin

7.11.4.69 GPDCR Register (Offset = FEh) [reset = 0h]

GPDCR is shown in [Figure 7-120](#) and described in [Table 7-131](#).

GPIO D Lock Commit Register (GPIO96 to 127)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-120. GPDCR Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
R/WOnce-0h	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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-131. GPDCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/WOnce	0h	Configuration lock commit bit for this pin
30	GPIO126	R/WOnce	0h	Configuration lock commit bit for this pin
29	GPIO125	R/WOnce	0h	Configuration lock commit bit for this pin
28	GPIO124	R/WOnce	0h	Configuration lock commit bit for this pin
27	GPIO123	R/WOnce	0h	Configuration lock commit bit for this pin
26	GPIO122	R/WOnce	0h	Configuration lock commit bit for this pin
25	GPIO121	R/WOnce	0h	Configuration lock commit bit for this pin
24	GPIO120	R/WOnce	0h	Configuration lock commit bit for this pin
23	GPIO119	R/WOnce	0h	Configuration lock commit bit for this pin
22	GPIO118	R/WOnce	0h	Configuration lock commit bit for this pin
21	GPIO117	R/WOnce	0h	Configuration lock commit bit for this pin
20	GPIO116	R/WOnce	0h	Configuration lock commit bit for this pin
19	GPIO115	R/WOnce	0h	Configuration lock commit bit for this pin
18	GPIO114	R/WOnce	0h	Configuration lock commit bit for this pin
17	GPIO113	R/WOnce	0h	Configuration lock commit bit for this pin
16	GPIO112	R/WOnce	0h	Configuration lock commit bit for this pin
15	GPIO111	R/WOnce	0h	Configuration lock commit bit for this pin
14	GPIO110	R/WOnce	0h	Configuration lock commit bit for this pin
13	GPIO109	R/WOnce	0h	Configuration lock commit bit for this pin
12	GPIO108	R/WOnce	0h	Configuration lock commit bit for this pin
11	GPIO107	R/WOnce	0h	Configuration lock commit bit for this pin
10	GPIO106	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-131. GPDCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO105	R/WOnce	0h	Configuration lock commit bit for this pin
8	GPIO104	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO103	R/WOnce	0h	Configuration lock commit bit for this pin
6	GPIO102	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO101	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO100	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO99	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO98	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO97	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO96	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.4.70 GPECTRL Register (Offset = 100h) [reset = 0h]

GPECTRL is shown in [Figure 7-121](#) and described in [Table 7-132](#).

GPIO E Qualification Sampling Period Control (GPIO128 to 159)

Figure 7-121. GPECTRL Register

31	30	29	28	27	26	25	24	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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-132. GPECTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	QUALPRD3	R/W	0h	Qualification sampling period for GPIO152 to GPIO159: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO144 to GPIO151: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO136 to GPIO143: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO128 to GPIO135: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 0xFF,QUALPRDx = PLLSYSCLK/510

7.11.4.71 GPEQSEL1 Register (Offset = 102h) [reset = 0h]

GPEQSEL1 is shown in [Figure 7-122](#) and described in [Table 7-133](#).

GPIO E Qualifier Select 1 Register (GPIO128 to 143)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-122. GPEQSEL1 Register

31	30	29	28	27	26	25	24
GPIO143		GPIO142		GPIO141		GPIO140	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO139		GPIO138		GPIO137		GPIO136	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO135		GPIO134		GPIO133		GPIO132	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO131		GPIO130		GPIO129		GPIO128	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-133. GPEQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO143	R/W	0h	Input qualification type
29-28	GPIO142	R/W	0h	Input qualification type
27-26	GPIO141	R/W	0h	Input qualification type
25-24	GPIO140	R/W	0h	Input qualification type
23-22	GPIO139	R/W	0h	Input qualification type
21-20	GPIO138	R/W	0h	Input qualification type
19-18	GPIO137	R/W	0h	Input qualification type
17-16	GPIO136	R/W	0h	Input qualification type
15-14	GPIO135	R/W	0h	Input qualification type
13-12	GPIO134	R/W	0h	Input qualification type
11-10	GPIO133	R/W	0h	Input qualification type
9-8	GPIO132	R/W	0h	Input qualification type
7-6	GPIO131	R/W	0h	Input qualification type
5-4	GPIO130	R/W	0h	Input qualification type
3-2	GPIO129	R/W	0h	Input qualification type
1-0	GPIO128	R/W	0h	Input qualification type

7.11.4.72 GPEQSEL2 Register (Offset = 104h) [reset = 0h]

GPEQSEL2 is shown in [Figure 7-123](#) and described in [Table 7-134](#).

GPIO E Qualifier Select 2 Register (GPIO144 to 159)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-123. GPEQSEL2 Register

31	30	29	28	27	26	25	24
GPIO159		GPIO158		GPIO157		GPIO156	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO155		GPIO154		GPIO153		GPIO152	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO151		GPIO150		GPIO149		GPIO148	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO147		GPIO146		GPIO145		GPIO144	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-134. GPEQSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO159	R/W	0h	Input qualification type
29-28	GPIO158	R/W	0h	Input qualification type
27-26	GPIO157	R/W	0h	Input qualification type
25-24	GPIO156	R/W	0h	Input qualification type
23-22	GPIO155	R/W	0h	Input qualification type
21-20	GPIO154	R/W	0h	Input qualification type
19-18	GPIO153	R/W	0h	Input qualification type
17-16	GPIO152	R/W	0h	Input qualification type
15-14	GPIO151	R/W	0h	Input qualification type
13-12	GPIO150	R/W	0h	Input qualification type
11-10	GPIO149	R/W	0h	Input qualification type
9-8	GPIO148	R/W	0h	Input qualification type
7-6	GPIO147	R/W	0h	Input qualification type
5-4	GPIO146	R/W	0h	Input qualification type
3-2	GPIO145	R/W	0h	Input qualification type
1-0	GPIO144	R/W	0h	Input qualification type

7.11.4.73 GPOMUX1 Register (Offset = 106h) [reset = 0h]

GPOMUX1 is shown in [Figure 7-124](#) and described in [Table 7-135](#).

GPIO E Mux 1 Register (GPIO128 to 143)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-124. GPOMUX1 Register

31	30	29	28	27	26	25	24
GPIO143		GPIO142		GPIO141		GPIO140	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO139		GPIO138		GPIO137		GPIO136	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO135		GPIO134		GPIO133		GPIO132	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO131		GPIO130		GPIO129		GPIO128	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-135. GPOMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO143	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO142	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO141	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO140	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO139	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO138	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO137	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO136	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO135	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO134	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO133	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO132	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO131	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO130	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO129	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO128	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.74 GPOMUX2 Register (Offset = 108h) [reset = 0h]

GPOMUX2 is shown in [Figure 7-125](#) and described in [Table 7-136](#).

GPIO E Mux 2 Register (GPIO144 to 159)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-125. GPOMUX2 Register

31	30	29	28	27	26	25	24
GPIO159		GPIO158		GPIO157		GPIO156	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO155		GPIO154		GPIO153		GPIO152	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO151		GPIO150		GPIO149		GPIO148	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO147		GPIO146		GPIO145		GPIO144	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-136. GPOMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO159	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO158	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO157	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO156	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO155	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO154	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO153	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO152	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO151	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO150	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO149	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO148	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO147	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO146	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO145	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO144	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.75 GPEDIR Register (Offset = 10Ah) [reset = 0h]

GPEDIR is shown in [Figure 7-126](#) and described in [Table 7-137](#).

GPIO E Direction Register (GPIO128 to 159)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-126. GPEDIR Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-137. GPEDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Defines direction for this pin in GPIO mode
30	GPIO158	R/W	0h	Defines direction for this pin in GPIO mode
29	GPIO157	R/W	0h	Defines direction for this pin in GPIO mode
28	GPIO156	R/W	0h	Defines direction for this pin in GPIO mode
27	GPIO155	R/W	0h	Defines direction for this pin in GPIO mode
26	GPIO154	R/W	0h	Defines direction for this pin in GPIO mode
25	GPIO153	R/W	0h	Defines direction for this pin in GPIO mode
24	GPIO152	R/W	0h	Defines direction for this pin in GPIO mode
23	GPIO151	R/W	0h	Defines direction for this pin in GPIO mode
22	GPIO150	R/W	0h	Defines direction for this pin in GPIO mode
21	GPIO149	R/W	0h	Defines direction for this pin in GPIO mode
20	GPIO148	R/W	0h	Defines direction for this pin in GPIO mode
19	GPIO147	R/W	0h	Defines direction for this pin in GPIO mode
18	GPIO146	R/W	0h	Defines direction for this pin in GPIO mode
17	GPIO145	R/W	0h	Defines direction for this pin in GPIO mode
16	GPIO144	R/W	0h	Defines direction for this pin in GPIO mode
15	GPIO143	R/W	0h	Defines direction for this pin in GPIO mode
14	GPIO142	R/W	0h	Defines direction for this pin in GPIO mode
13	GPIO141	R/W	0h	Defines direction for this pin in GPIO mode
12	GPIO140	R/W	0h	Defines direction for this pin in GPIO mode
11	GPIO139	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-137. GPEDIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO138	R/W	0h	Defines direction for this pin in GPIO mode
9	GPIO137	R/W	0h	Defines direction for this pin in GPIO mode
8	GPIO136	R/W	0h	Defines direction for this pin in GPIO mode
7	GPIO135	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO134	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO133	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO132	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO131	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO130	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO129	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO128	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.76 GPEPUD Register (Offset = 10Ch) [reset = FFFFFFFFh]

GPEPUD is shown in [Figure 7-127](#) and described in [Table 7-138](#).

GPIO E Pull Up Disable Register (GPIO128 to 159)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-127. GPEPUD Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-138. GPEPUD Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	1h	Pull-Up Disable control for this pin
30	GPIO158	R/W	1h	Pull-Up Disable control for this pin
29	GPIO157	R/W	1h	Pull-Up Disable control for this pin
28	GPIO156	R/W	1h	Pull-Up Disable control for this pin
27	GPIO155	R/W	1h	Pull-Up Disable control for this pin
26	GPIO154	R/W	1h	Pull-Up Disable control for this pin
25	GPIO153	R/W	1h	Pull-Up Disable control for this pin
24	GPIO152	R/W	1h	Pull-Up Disable control for this pin
23	GPIO151	R/W	1h	Pull-Up Disable control for this pin
22	GPIO150	R/W	1h	Pull-Up Disable control for this pin
21	GPIO149	R/W	1h	Pull-Up Disable control for this pin
20	GPIO148	R/W	1h	Pull-Up Disable control for this pin
19	GPIO147	R/W	1h	Pull-Up Disable control for this pin
18	GPIO146	R/W	1h	Pull-Up Disable control for this pin
17	GPIO145	R/W	1h	Pull-Up Disable control for this pin
16	GPIO144	R/W	1h	Pull-Up Disable control for this pin
15	GPIO143	R/W	1h	Pull-Up Disable control for this pin
14	GPIO142	R/W	1h	Pull-Up Disable control for this pin
13	GPIO141	R/W	1h	Pull-Up Disable control for this pin
12	GPIO140	R/W	1h	Pull-Up Disable control for this pin
11	GPIO139	R/W	1h	Pull-Up Disable control for this pin

Table 7-138. GPEPUD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO138	R/W	1h	Pull-Up Disable control for this pin
9	GPIO137	R/W	1h	Pull-Up Disable control for this pin
8	GPIO136	R/W	1h	Pull-Up Disable control for this pin
7	GPIO135	R/W	1h	Pull-Up Disable control for this pin
6	GPIO134	R/W	1h	Pull-Up Disable control for this pin
5	GPIO133	R/W	1h	Pull-Up Disable control for this pin
4	GPIO132	R/W	1h	Pull-Up Disable control for this pin
3	GPIO131	R/W	1h	Pull-Up Disable control for this pin
2	GPIO130	R/W	1h	Pull-Up Disable control for this pin
1	GPIO129	R/W	1h	Pull-Up Disable control for this pin
0	GPIO128	R/W	1h	Pull-Up Disable control for this pin

7.11.4.77 GPEINV Register (Offset = 110h) [reset = 0h]

GPEINV is shown in [Figure 7-128](#) and described in [Table 7-139](#).

GPIO E Input Polarity Invert Registers (GPIO128 to 159)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-128. GPEINV Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-139. GPEINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Input inversion control for this pin
30	GPIO158	R/W	0h	Input inversion control for this pin
29	GPIO157	R/W	0h	Input inversion control for this pin
28	GPIO156	R/W	0h	Input inversion control for this pin
27	GPIO155	R/W	0h	Input inversion control for this pin
26	GPIO154	R/W	0h	Input inversion control for this pin
25	GPIO153	R/W	0h	Input inversion control for this pin
24	GPIO152	R/W	0h	Input inversion control for this pin
23	GPIO151	R/W	0h	Input inversion control for this pin
22	GPIO150	R/W	0h	Input inversion control for this pin
21	GPIO149	R/W	0h	Input inversion control for this pin
20	GPIO148	R/W	0h	Input inversion control for this pin
19	GPIO147	R/W	0h	Input inversion control for this pin
18	GPIO146	R/W	0h	Input inversion control for this pin
17	GPIO145	R/W	0h	Input inversion control for this pin
16	GPIO144	R/W	0h	Input inversion control for this pin
15	GPIO143	R/W	0h	Input inversion control for this pin
14	GPIO142	R/W	0h	Input inversion control for this pin
13	GPIO141	R/W	0h	Input inversion control for this pin
12	GPIO140	R/W	0h	Input inversion control for this pin
11	GPIO139	R/W	0h	Input inversion control for this pin

Table 7-139. GPEINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	GPIO138	R/W	0h	Input inversion control for this pin
9	GPIO137	R/W	0h	Input inversion control for this pin
8	GPIO136	R/W	0h	Input inversion control for this pin
7	GPIO135	R/W	0h	Input inversion control for this pin
6	GPIO134	R/W	0h	Input inversion control for this pin
5	GPIO133	R/W	0h	Input inversion control for this pin
4	GPIO132	R/W	0h	Input inversion control for this pin
3	GPIO131	R/W	0h	Input inversion control for this pin
2	GPIO130	R/W	0h	Input inversion control for this pin
1	GPIO129	R/W	0h	Input inversion control for this pin
0	GPIO128	R/W	0h	Input inversion control for this pin

7.11.4.78 GPEODR Register (Offset = 112h) [reset = 0h]

GPEODR is shown in [Figure 7-129](#) and described in [Table 7-140](#).

GPIO E Open Drain Output Register (GPIO128 to GPIO159)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-129. GPEODR Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-140. GPEODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Output Open-Drain control for this pin
30	GPIO158	R/W	0h	Output Open-Drain control for this pin
29	GPIO157	R/W	0h	Output Open-Drain control for this pin
28	GPIO156	R/W	0h	Output Open-Drain control for this pin
27	GPIO155	R/W	0h	Output Open-Drain control for this pin
26	GPIO154	R/W	0h	Output Open-Drain control for this pin
25	GPIO153	R/W	0h	Output Open-Drain control for this pin
24	GPIO152	R/W	0h	Output Open-Drain control for this pin
23	GPIO151	R/W	0h	Output Open-Drain control for this pin
22	GPIO150	R/W	0h	Output Open-Drain control for this pin
21	GPIO149	R/W	0h	Output Open-Drain control for this pin
20	GPIO148	R/W	0h	Output Open-Drain control for this pin
19	GPIO147	R/W	0h	Output Open-Drain control for this pin
18	GPIO146	R/W	0h	Output Open-Drain control for this pin
17	GPIO145	R/W	0h	Output Open-Drain control for this pin
16	GPIO144	R/W	0h	Output Open-Drain control for this pin
15	GPIO143	R/W	0h	Output Open-Drain control for this pin
14	GPIO142	R/W	0h	Output Open-Drain control for this pin
13	GPIO141	R/W	0h	Output Open-Drain control for this pin

Table 7-140. GPEODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	GPIO140	R/W	0h	Output Open-Drain control for this pin
11	GPIO139	R/W	0h	Output Open-Drain control for this pin
10	GPIO138	R/W	0h	Output Open-Drain control for this pin
9	GPIO137	R/W	0h	Output Open-Drain control for this pin
8	GPIO136	R/W	0h	Output Open-Drain control for this pin
7	GPIO135	R/W	0h	Output Open-Drain control for this pin
6	GPIO134	R/W	0h	Output Open-Drain control for this pin
5	GPIO133	R/W	0h	Output Open-Drain control for this pin
4	GPIO132	R/W	0h	Output Open-Drain control for this pin
3	GPIO131	R/W	0h	Output Open-Drain control for this pin
2	GPIO130	R/W	0h	Output Open-Drain control for this pin
1	GPIO129	R/W	0h	Output Open-Drain control for this pin
0	GPIO128	R/W	0h	Output Open-Drain control for this pin

7.11.4.79 GPEGMUX1 Register (Offset = 120h) [reset = 0h]

GPEGMUX1 is shown in [Figure 7-130](#) and described in [Table 7-141](#).

GPIO E Peripheral Group Mux (GPIO128 to 143)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-130. GPEGMUX1 Register

31	30	29	28	27	26	25	24
GPIO143		GPIO142		GPIO141		GPIO140	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO139		GPIO138		GPIO137		GPIO136	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO135		GPIO134		GPIO133		GPIO132	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO131		GPIO130		GPIO129		GPIO128	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-141. GPEGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO143	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO142	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO141	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO140	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO139	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO138	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO137	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO136	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO135	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO134	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO133	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO132	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO131	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO130	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO129	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO128	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.80 GPEGMUX2 Register (Offset = 122h) [reset = 0h]

GPEGMUX2 is shown in [Figure 7-131](#) and described in [Table 7-142](#).

GPIO E Peripheral Group Mux (GPIO144 to 159)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-131. GPEGMUX2 Register

31	30	29	28	27	26	25	24
GPIO159		GPIO158		GPIO157		GPIO156	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
GPIO155		GPIO154		GPIO153		GPIO152	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO151		GPIO150		GPIO149		GPIO148	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO147		GPIO146		GPIO145		GPIO144	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-142. GPEGMUX2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	GPIO159	R/W	0h	Defines pin-muxing selection for GPIO
29-28	GPIO158	R/W	0h	Defines pin-muxing selection for GPIO
27-26	GPIO157	R/W	0h	Defines pin-muxing selection for GPIO
25-24	GPIO156	R/W	0h	Defines pin-muxing selection for GPIO
23-22	GPIO155	R/W	0h	Defines pin-muxing selection for GPIO
21-20	GPIO154	R/W	0h	Defines pin-muxing selection for GPIO
19-18	GPIO153	R/W	0h	Defines pin-muxing selection for GPIO
17-16	GPIO152	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO151	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO150	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO149	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO148	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO147	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO146	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO145	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO144	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.81 GPECSEL1 Register (Offset = 128h) [reset = 0h]

GPECSEL1 is shown in [Figure 7-132](#) and described in [Table 7-143](#).

GPIO E Core Select Register (GPIO128 to 135)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-132. GPECSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO135				GPIO134				GPIO133				GPIO132			
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
GPIO131				GPIO130				GPIO129				GPIO128			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-143. GPECSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO135	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO134	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO133	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO132	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO131	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO130	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO129	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO128	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.82 GPECSEL2 Register (Offset = 12Ah) [reset = 0h]

GPECSEL2 is shown in [Figure 7-133](#) and described in [Table 7-144](#).

GPIO E Core Select Register (GPIO136 to 143)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-133. GPECSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO143				GPIO142				GPIO141				GPIO140			
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
GPIO139				GPIO138				GPIO137				GPIO136			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-144. GPECSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO143	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO142	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO141	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO140	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO139	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO138	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO137	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO136	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.83 GPECSEL3 Register (Offset = 12Ch) [reset = 0h]

GPECSEL3 is shown in [Figure 7-134](#) and described in [Table 7-145](#).

GPIO E Core Select Register (GPIO144 to 151)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-134. GPECSEL3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO151				GPIO150				GPIO149				GPIO148			
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
GPIO147				GPIO146				GPIO145				GPIO144			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-145. GPECSEL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO151	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO150	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO149	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO148	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO147	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO146	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO145	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO144	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.84 GPECSEL4 Register (Offset = 12Eh) [reset = 0h]

GPECSEL4 is shown in [Figure 7-135](#) and described in [Table 7-146](#).

GPIO E Core Select Register (GPIO152 to 159)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-135. GPECSEL4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO159				GPIO158				GPIO157				GPIO156			
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
GPIO155				GPIO154				GPIO153				GPIO152			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-146. GPECSEL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO159	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO158	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO157	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO156	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO155	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO154	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO153	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO152	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.85 GPELOCK Register (Offset = 13Ch) [reset = 0h]

GPELOCK is shown in [Figure 7-136](#) and described in [Table 7-147](#).

GPIO E Lock Configuration Register (GPIO128 to 159)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-136. GPELOCK Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-147. GPELOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Configuration Lock bit for this pin
30	GPIO158	R/W	0h	Configuration Lock bit for this pin
29	GPIO157	R/W	0h	Configuration Lock bit for this pin
28	GPIO156	R/W	0h	Configuration Lock bit for this pin
27	GPIO155	R/W	0h	Configuration Lock bit for this pin
26	GPIO154	R/W	0h	Configuration Lock bit for this pin
25	GPIO153	R/W	0h	Configuration Lock bit for this pin
24	GPIO152	R/W	0h	Configuration Lock bit for this pin
23	GPIO151	R/W	0h	Configuration Lock bit for this pin
22	GPIO150	R/W	0h	Configuration Lock bit for this pin
21	GPIO149	R/W	0h	Configuration Lock bit for this pin
20	GPIO148	R/W	0h	Configuration Lock bit for this pin
19	GPIO147	R/W	0h	Configuration Lock bit for this pin
18	GPIO146	R/W	0h	Configuration Lock bit for this pin
17	GPIO145	R/W	0h	Configuration Lock bit for this pin
16	GPIO144	R/W	0h	Configuration Lock bit for this pin
15	GPIO143	R/W	0h	Configuration Lock bit for this pin
14	GPIO142	R/W	0h	Configuration Lock bit for this pin
13	GPIO141	R/W	0h	Configuration Lock bit for this pin
12	GPIO140	R/W	0h	Configuration Lock bit for this pin

Table 7-147. GPELOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	GPIO139	R/W	0h	Configuration Lock bit for this pin
10	GPIO138	R/W	0h	Configuration Lock bit for this pin
9	GPIO137	R/W	0h	Configuration Lock bit for this pin
8	GPIO136	R/W	0h	Configuration Lock bit for this pin
7	GPIO135	R/W	0h	Configuration Lock bit for this pin
6	GPIO134	R/W	0h	Configuration Lock bit for this pin
5	GPIO133	R/W	0h	Configuration Lock bit for this pin
4	GPIO132	R/W	0h	Configuration Lock bit for this pin
3	GPIO131	R/W	0h	Configuration Lock bit for this pin
2	GPIO130	R/W	0h	Configuration Lock bit for this pin
1	GPIO129	R/W	0h	Configuration Lock bit for this pin
0	GPIO128	R/W	0h	Configuration Lock bit for this pin

7.11.4.86 GPECR Register (Offset = 13Eh) [reset = 0h]

GPECR is shown in [Figure 7-137](#) and described in [Table 7-148](#).

GPIO E Lock Commit Register (GPIO128 to 159)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-137. GPECR Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
R/WOnce-0h	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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-148. GPECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/WOnce	0h	Configuration lock commit bit for this pin
30	GPIO158	R/WOnce	0h	Configuration lock commit bit for this pin
29	GPIO157	R/WOnce	0h	Configuration lock commit bit for this pin
28	GPIO156	R/WOnce	0h	Configuration lock commit bit for this pin
27	GPIO155	R/WOnce	0h	Configuration lock commit bit for this pin
26	GPIO154	R/WOnce	0h	Configuration lock commit bit for this pin
25	GPIO153	R/WOnce	0h	Configuration lock commit bit for this pin
24	GPIO152	R/WOnce	0h	Configuration lock commit bit for this pin
23	GPIO151	R/WOnce	0h	Configuration lock commit bit for this pin
22	GPIO150	R/WOnce	0h	Configuration lock commit bit for this pin
21	GPIO149	R/WOnce	0h	Configuration lock commit bit for this pin
20	GPIO148	R/WOnce	0h	Configuration lock commit bit for this pin
19	GPIO147	R/WOnce	0h	Configuration lock commit bit for this pin
18	GPIO146	R/WOnce	0h	Configuration lock commit bit for this pin
17	GPIO145	R/WOnce	0h	Configuration lock commit bit for this pin
16	GPIO144	R/WOnce	0h	Configuration lock commit bit for this pin
15	GPIO143	R/WOnce	0h	Configuration lock commit bit for this pin
14	GPIO142	R/WOnce	0h	Configuration lock commit bit for this pin
13	GPIO141	R/WOnce	0h	Configuration lock commit bit for this pin
12	GPIO140	R/WOnce	0h	Configuration lock commit bit for this pin
11	GPIO139	R/WOnce	0h	Configuration lock commit bit for this pin
10	GPIO138	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-148. GPECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO137	R/WOnce	0h	Configuration lock commit bit for this pin
8	GPIO136	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO135	R/WOnce	0h	Configuration lock commit bit for this pin
6	GPIO134	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO133	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO132	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO131	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO130	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO129	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO128	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.4.87 GPFCTRL Register (Offset = 140h) [reset = 0h]

GPFCTRL is shown in [Figure 7-138](#) and described in [Table 7-149](#).

GPIO F Qualification Sampling Period Control (GPIO160 to 168)

Figure 7-138. GPFCTRL Register

31	30	29	28	27	26	25	24	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								QUALPRD1								QUALPRD0							
R-0h								R-0h								R/W-0h								R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-149. GPFCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO168: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO160 to GPIO167: 0x00, QUALPRDx = PLLSYSCLK 0x01, QUALPRDx = PLLSYSCLK/2 0x02, QUALPRDx = PLLSYSCLK/4 0xFF, QUALPRDx = PLLSYSCLK/510

7.11.4.88 GPFQSEL1 Register (Offset = 142h) [reset = 0h]

GPFQSEL1 is shown in [Figure 7-139](#) and described in [Table 7-150](#).

GPIO F Qualifier Select 1 Register (GPIO160 to 168)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

Figure 7-139. GPFQSEL1 Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
15	14	13	12	11	10	9	8
GPIO167	GPIO167	GPIO166	GPIO166	GPIO165	GPIO165	GPIO164	GPIO164
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
GPIO163	GPIO163	GPIO162	GPIO162	GPIO161	GPIO161	GPIO160	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-150. GPFQSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	RESERVED	R	0h	Reserved
27-26	RESERVED	R	0h	Reserved
25-24	RESERVED	R	0h	Reserved
23-22	RESERVED	R	0h	Reserved
21-20	RESERVED	R	0h	Reserved
19-18	RESERVED	R	0h	Reserved
17-16	GPIO168	R/W	0h	Input qualification type
15-14	GPIO167	R/W	0h	Input qualification type
13-12	GPIO166	R/W	0h	Input qualification type
11-10	GPIO165	R/W	0h	Input qualification type
9-8	GPIO164	R/W	0h	Input qualification type
7-6	GPIO163	R/W	0h	Input qualification type
5-4	GPIO162	R/W	0h	Input qualification type
3-2	GPIO161	R/W	0h	Input qualification type
1-0	GPIO160	R/W	0h	Input qualification type

7.11.4.89 GPFMUX1 Register (Offset = 146h) [reset = 0h]

GPFMUX1 is shown in [Figure 7-140](#) and described in [Table 7-151](#).

GPIO F Mux 1 Register (GPIO160 to 168)

Defines pin-muxing selection for GPIO.

Notes:

The respective GPyGMUXn.GPIOz must be configured prior to this register to avoid intermediate peripheral selects being mapped to the GPIO.

Figure 7-140. GPFMUX1 Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
15	14	13	12	11	10	9	8
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
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
GPIO163	GPIO162	GPIO161	GPIO160	GPIO159	GPIO158	GPIO157	GPIO156
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-151. GPFMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	RESERVED	R	0h	Reserved
27-26	RESERVED	R	0h	Reserved
25-24	RESERVED	R	0h	Reserved
23-22	RESERVED	R	0h	Reserved
21-20	RESERVED	R	0h	Reserved
19-18	RESERVED	R	0h	Reserved
17-16	GPIO168	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO167	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO166	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO165	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO164	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO163	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO162	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO161	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO160	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.90 GPFDIR Register (Offset = 14Ah) [reset = 0h]

GPFDIR is shown in [Figure 7-141](#) and described in [Table 7-152](#).

GPIO F Direction Register (GPIO160 to 168)

Controls direction of GPIO pins when the specified pin is configured in GPIO mode.

0: Configures pin as input.

1: Configures pin as output.

Reading the register returns the current value of the register setting.

Figure 7-141. GPFDIR Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-152. GPFDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Defines direction for this pin in GPIO mode

Table 7-152. GPFDIR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	GPIO167	R/W	0h	Defines direction for this pin in GPIO mode
6	GPIO166	R/W	0h	Defines direction for this pin in GPIO mode
5	GPIO165	R/W	0h	Defines direction for this pin in GPIO mode
4	GPIO164	R/W	0h	Defines direction for this pin in GPIO mode
3	GPIO163	R/W	0h	Defines direction for this pin in GPIO mode
2	GPIO162	R/W	0h	Defines direction for this pin in GPIO mode
1	GPIO161	R/W	0h	Defines direction for this pin in GPIO mode
0	GPIO160	R/W	0h	Defines direction for this pin in GPIO mode

7.11.4.91 GPFPU Register (Offset = 14Ch) [reset = FFFFFFFFh]

GPFPU is shown in [Figure 7-142](#) and described in [Table 7-153](#).

GPIO F Pull Up Disable Register (GPIO160 to 168)

Disables the Pull-Up on GPIO.

0: Enables the Pull-Up.

1: Disables the Pull-Up.

Reading the register returns the current value of the register setting.

Figure 7-142. GPFPU Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-1h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-153. GPFPU Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	1h	Pull-Up Disable control for this pin

Table 7-153. GPFPU Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	GPIO167	R/W	1h	Pull-Up Disable control for this pin
6	GPIO166	R/W	1h	Pull-Up Disable control for this pin
5	GPIO165	R/W	1h	Pull-Up Disable control for this pin
4	GPIO164	R/W	1h	Pull-Up Disable control for this pin
3	GPIO163	R/W	1h	Pull-Up Disable control for this pin
2	GPIO162	R/W	1h	Pull-Up Disable control for this pin
1	GPIO161	R/W	1h	Pull-Up Disable control for this pin
0	GPIO160	R/W	1h	Pull-Up Disable control for this pin

7.11.4.92 GPFINV Register (Offset = 150h) [reset = 0h]

GPFINV is shown in [Figure 7-143](#) and described in [Table 7-154](#).

GPIO F Input Polarity Invert Registers (GPIO160 to 168)

Selects between non-inverted and inverted GPIO input to the device.

0: selects non-inverted GPIO input

1: selects inverted GPIO input

Reading the register returns the current value of the register setting.

Figure 7-143. GPFINV Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-154. GPFINV Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Input inversion control for this pin

Table 7-154. GPFINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	GPIO167	R/W	0h	Input inversion control for this pin
6	GPIO166	R/W	0h	Input inversion control for this pin
5	GPIO165	R/W	0h	Input inversion control for this pin
4	GPIO164	R/W	0h	Input inversion control for this pin
3	GPIO163	R/W	0h	Input inversion control for this pin
2	GPIO162	R/W	0h	Input inversion control for this pin
1	GPIO161	R/W	0h	Input inversion control for this pin
0	GPIO160	R/W	0h	Input inversion control for this pin

7.11.4.93 GPFODR Register (Offset = 152h) [reset = 0h]

GPFODR is shown in [Figure 7-144](#) and described in [Table 7-155](#).

GPIO F Open Drain Output Register (GPIO160 to GPIO168)

Selects between normal and open-drain output for the GPIO pin.

0: Normal Output

1: Open Drain Output

Reading the register returns the current value of the register setting.

Note:

[1] In the Open Drain output mode, if the buffer is configured for output mode, a 0 value to be driven out comes out on the on the PAD while a 1 value to be driven out tri-states the buffer.

Figure 7-144. GPFODR Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-155. GPFODR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved

Table 7-155. GPFODR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Output Open-Drain control for this pin
7	GPIO167	R/W	0h	Output Open-Drain control for this pin
6	GPIO166	R/W	0h	Output Open-Drain control for this pin
5	GPIO165	R/W	0h	Output Open-Drain control for this pin
4	GPIO164	R/W	0h	Output Open-Drain control for this pin
3	GPIO163	R/W	0h	Output Open-Drain control for this pin
2	GPIO162	R/W	0h	Output Open-Drain control for this pin
1	GPIO161	R/W	0h	Output Open-Drain control for this pin
0	GPIO160	R/W	0h	Output Open-Drain control for this pin

7.11.4.94 GPFGMUX1 Register (Offset = 160h) [reset = 0h]

GPFGMUX1 is shown in [Figure 7-145](#) and described in [Table 7-156](#).

GPIO F Peripheral Group Mux (GPIO160 to 168)

Defines pin-muxing selection for GPIO.

Note: For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

Figure 7-145. GPFGMUX1 Register

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		GPIO168	
R-0h		R-0h		R-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO167		GPIO166		GPIO165		GPIO164	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO163		GPIO162		GPIO161		GPIO160	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-156. GPFGMUX1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-28	RESERVED	R	0h	Reserved
27-26	RESERVED	R	0h	Reserved
25-24	RESERVED	R	0h	Reserved
23-22	RESERVED	R	0h	Reserved
21-20	RESERVED	R	0h	Reserved
19-18	RESERVED	R	0h	Reserved
17-16	GPIO168	R/W	0h	Defines pin-muxing selection for GPIO
15-14	GPIO167	R/W	0h	Defines pin-muxing selection for GPIO
13-12	GPIO166	R/W	0h	Defines pin-muxing selection for GPIO
11-10	GPIO165	R/W	0h	Defines pin-muxing selection for GPIO
9-8	GPIO164	R/W	0h	Defines pin-muxing selection for GPIO
7-6	GPIO163	R/W	0h	Defines pin-muxing selection for GPIO
5-4	GPIO162	R/W	0h	Defines pin-muxing selection for GPIO
3-2	GPIO161	R/W	0h	Defines pin-muxing selection for GPIO
1-0	GPIO160	R/W	0h	Defines pin-muxing selection for GPIO

7.11.4.95 GPFCSSEL1 Register (Offset = 168h) [reset = 0h]

GPFCSSEL1 is shown in [Figure 7-146](#) and described in [Table 7-157](#).

GPIO F Core Select Register (GPIO160 to 167)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-146. GPFCSSEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO167				GPIO166				GPIO165				GPIO164			
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
GPIO163				GPIO162				GPIO161				GPIO160			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-157. GPFCSSEL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	GPIO167	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
27-24	GPIO166	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
23-20	GPIO165	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
19-16	GPIO164	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
15-12	GPIO163	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
11-8	GPIO162	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
7-4	GPIO161	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin
3-0	GPIO160	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.96 GPFCSEL2 Register (Offset = 16Ah) [reset = 0h]

GPFCSEL2 is shown in [Figure 7-147](#) and described in [Table 7-158](#).

GPIO F Core Select Register (GPIO168)

Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

xx00: CPU1 selected

xx01: CPU1.CLA1 selected

xx10: CPU2 selected

xx11: CPU2.CLA1 selected

Figure 7-147. GPFCSEL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				RESERVED				RESERVED				RESERVED			
R-0h				R-0h				R-0h				R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				RESERVED				RESERVED				GPIO168			
R-0h				R-0h				R-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-158. GPFCSEL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	RESERVED	R	0h	Reserved
23-20	RESERVED	R	0h	Reserved
19-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R	0h	Reserved
7-4	RESERVED	R	0h	Reserved
3-0	GPIO168	R/W	0h	Selects which master's GPIODAT/SET/CLEAR/TOGGLE registers control this GPIO pin

7.11.4.97 GPFLOCK Register (Offset = 17Ch) [reset = 0h]

GPFLOCK is shown in [Figure 7-148](#) and described in [Table 7-159](#).

GPIO F Lock Configuration Register (GPIO160 to 168)

GPIO Configuration Lock for GPIO.

0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed

1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin

Figure 7-148. GPFLOCK Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-159. GPFLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Configuration Lock bit for this pin

Table 7-159. GPFLOCK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	GPIO167	R/W	0h	Configuration Lock bit for this pin
6	GPIO166	R/W	0h	Configuration Lock bit for this pin
5	GPIO165	R/W	0h	Configuration Lock bit for this pin
4	GPIO164	R/W	0h	Configuration Lock bit for this pin
3	GPIO163	R/W	0h	Configuration Lock bit for this pin
2	GPIO162	R/W	0h	Configuration Lock bit for this pin
1	GPIO161	R/W	0h	Configuration Lock bit for this pin
0	GPIO160	R/W	0h	Configuration Lock bit for this pin

7.11.4.98 GPFCR Register (Offset = 17Eh) [reset = 0h]

GPFCR is shown in [Figure 7-149](#) and described in [Table 7-160](#).

GPIO F Lock Commit Register (GPIO160 to 168)

GPIO Configuration Lock Commit for GPIO:

1: Locks changes to the bit in GPyLOCK register which controls the same pin

0: Bit in the GPyLOCK register which controls the same pin can be changed

Figure 7-149. GPFCR Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-160. GPFCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/WOnce	0h	Configuration lock commit bit for this pin
7	GPIO167	R/WOnce	0h	Configuration lock commit bit for this pin

Table 7-160. GPFCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	GPIO166	R/WOnce	0h	Configuration lock commit bit for this pin
5	GPIO165	R/WOnce	0h	Configuration lock commit bit for this pin
4	GPIO164	R/WOnce	0h	Configuration lock commit bit for this pin
3	GPIO163	R/WOnce	0h	Configuration lock commit bit for this pin
2	GPIO162	R/WOnce	0h	Configuration lock commit bit for this pin
1	GPIO161	R/WOnce	0h	Configuration lock commit bit for this pin
0	GPIO160	R/WOnce	0h	Configuration lock commit bit for this pin

7.11.5 GPIO_DATA_REGS Registers

Table 7-161 lists the memory-mapped registers for the GPIO_DATA_REGS. All register offset addresses not listed in Table 7-161 should be considered as reserved locations and the register contents should not be modified.

Table 7-161. GPIO_DATA_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	GPADAT	GPIO A Data Register (GPIO0 to 31)		Go
2h	GPASET	GPIO A Data Set Register (GPIO0 to 31)		Go
4h	GPACLEAR	GPIO A Data Clear Register (GPIO0 to 31)		Go
6h	GPATOGGLE	GPIO A Data Toggle Register (GPIO0 to 31)		Go
8h	GPBDAT	GPIO B Data Register (GPIO32 to 63)		Go
Ah	GPBSET	GPIO B Data Set Register (GPIO32 to 63)		Go
Ch	GPBCLEAR	GPIO B Data Clear Register (GPIO32 to 63)		Go
Eh	GPBTOGGLE	GPIO B Data Toggle Register (GPIO32 to 63)		Go
10h	GPCDAT	GPIO C Data Register (GPIO64 to 95)		Go
12h	GPCSET	GPIO C Data Set Register (GPIO64 to 95)		Go
14h	GPCCLEAR	GPIO C Data Clear Register (GPIO64 to 95)		Go
16h	GPCTOGGLE	GPIO C Data Toggle Register (GPIO64 to 95)		Go
18h	GPDDAT	GPIO D Data Register (GPIO96 to 127)		Go
1Ah	GPDSET	GPIO D Data Set Register (GPIO96 to 127)		Go
1Ch	GPDCLEAR	GPIO D Data Clear Register (GPIO96 to 127)		Go
1Eh	GPDTOGGLE	GPIO D Data Toggle Register (GPIO96 to 127)		Go
20h	GPEDAT	GPIO E Data Register (GPIO128 to 159)		Go
22h	GPASET	GPIO E Data Set Register (GPIO128 to 159)		Go
24h	GPECLEAR	GPIO E Data Clear Register (GPIO128 to 159)		Go
26h	GPETOGGLE	GPIO E Data Toggle Register (GPIO128 to 159)		Go
28h	GPFDAT	GPIO F Data Register (GPIO160 to 168)		Go
2Ah	GPFSET	GPIO F Data Set Register (GPIO160 to 168)		Go
2Ch	GPFCLEAR	GPIO F Data Clear Register (GPIO160 to 168)		Go
2Eh	GPFTOGGLE	GPIO F Data Toggle Register (GPIO160 to 168)		Go

7.11.5.1 GPADAT Register (Offset = 0h) [reset = 0h]

GPADAT is shown in [Figure 7-150](#) and described in [Table 7-162](#).

GPIO A Data Register (GPIO0 to 31)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-150. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-162. GPADAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Data Register for this pin
30	GPIO30	R/W	0h	Data Register for this pin
29	GPIO29	R/W	0h	Data Register for this pin
28	GPIO28	R/W	0h	Data Register for this pin
27	GPIO27	R/W	0h	Data Register for this pin
26	GPIO26	R/W	0h	Data Register for this pin
25	GPIO25	R/W	0h	Data Register for this pin
24	GPIO24	R/W	0h	Data Register for this pin
23	GPIO23	R/W	0h	Data Register for this pin
22	GPIO22	R/W	0h	Data Register for this pin
21	GPIO21	R/W	0h	Data Register for this pin
20	GPIO20	R/W	0h	Data Register for this pin
19	GPIO19	R/W	0h	Data Register for this pin
18	GPIO18	R/W	0h	Data Register for this pin
17	GPIO17	R/W	0h	Data Register for this pin
16	GPIO16	R/W	0h	Data Register for this pin
15	GPIO15	R/W	0h	Data Register for this pin

Table 7-162. GPADAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	GPIO14	R/W	0h	Data Register for this pin
13	GPIO13	R/W	0h	Data Register for this pin
12	GPIO12	R/W	0h	Data Register for this pin
11	GPIO11	R/W	0h	Data Register for this pin
10	GPIO10	R/W	0h	Data Register for this pin
9	GPIO9	R/W	0h	Data Register for this pin
8	GPIO8	R/W	0h	Data Register for this pin
7	GPIO7	R/W	0h	Data Register for this pin
6	GPIO6	R/W	0h	Data Register for this pin
5	GPIO5	R/W	0h	Data Register for this pin
4	GPIO4	R/W	0h	Data Register for this pin
3	GPIO3	R/W	0h	Data Register for this pin
2	GPIO2	R/W	0h	Data Register for this pin
1	GPIO1	R/W	0h	Data Register for this pin
0	GPIO0	R/W	0h	Data Register for this pin

7.11.5.2 GPASET Register (Offset = 2h) [reset = 0h]

GPASET is shown in [Figure 7-151](#) and described in [Table 7-163](#).

GPIO A Data Set Register (GPIO0 to 31)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-151. GPASET 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-163. GPASET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output Set bit for this pin
30	GPIO30	R/W	0h	Output Set bit for this pin
29	GPIO29	R/W	0h	Output Set bit for this pin
28	GPIO28	R/W	0h	Output Set bit for this pin
27	GPIO27	R/W	0h	Output Set bit for this pin
26	GPIO26	R/W	0h	Output Set bit for this pin
25	GPIO25	R/W	0h	Output Set bit for this pin
24	GPIO24	R/W	0h	Output Set bit for this pin
23	GPIO23	R/W	0h	Output Set bit for this pin
22	GPIO22	R/W	0h	Output Set bit for this pin
21	GPIO21	R/W	0h	Output Set bit for this pin
20	GPIO20	R/W	0h	Output Set bit for this pin
19	GPIO19	R/W	0h	Output Set bit for this pin
18	GPIO18	R/W	0h	Output Set bit for this pin
17	GPIO17	R/W	0h	Output Set bit for this pin
16	GPIO16	R/W	0h	Output Set bit for this pin
15	GPIO15	R/W	0h	Output Set bit for this pin
14	GPIO14	R/W	0h	Output Set bit for this pin
13	GPIO13	R/W	0h	Output Set bit for this pin
12	GPIO12	R/W	0h	Output Set bit for this pin
11	GPIO11	R/W	0h	Output Set bit for this pin
10	GPIO10	R/W	0h	Output Set bit for this pin

Table 7-163. GPASET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO9	R/W	0h	Output Set bit for this pin
8	GPIO8	R/W	0h	Output Set bit for this pin
7	GPIO7	R/W	0h	Output Set bit for this pin
6	GPIO6	R/W	0h	Output Set bit for this pin
5	GPIO5	R/W	0h	Output Set bit for this pin
4	GPIO4	R/W	0h	Output Set bit for this pin
3	GPIO3	R/W	0h	Output Set bit for this pin
2	GPIO2	R/W	0h	Output Set bit for this pin
1	GPIO1	R/W	0h	Output Set bit for this pin
0	GPIO0	R/W	0h	Output Set bit for this pin

7.11.5.3 GPACLEAR Register (Offset = 4h) [reset = 0h]

GPACLEAR is shown in [Figure 7-152](#) and described in [Table 7-164](#).

GPIO A Data Clear Register (GPIO0 to 31)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-152. GPACLEAR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-164. GPACLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output Clear bit for this pin
30	GPIO30	R/W	0h	Output Clear bit for this pin
29	GPIO29	R/W	0h	Output Clear bit for this pin
28	GPIO28	R/W	0h	Output Clear bit for this pin
27	GPIO27	R/W	0h	Output Clear bit for this pin
26	GPIO26	R/W	0h	Output Clear bit for this pin
25	GPIO25	R/W	0h	Output Clear bit for this pin
24	GPIO24	R/W	0h	Output Clear bit for this pin
23	GPIO23	R/W	0h	Output Clear bit for this pin
22	GPIO22	R/W	0h	Output Clear bit for this pin
21	GPIO21	R/W	0h	Output Clear bit for this pin
20	GPIO20	R/W	0h	Output Clear bit for this pin
19	GPIO19	R/W	0h	Output Clear bit for this pin
18	GPIO18	R/W	0h	Output Clear bit for this pin
17	GPIO17	R/W	0h	Output Clear bit for this pin
16	GPIO16	R/W	0h	Output Clear bit for this pin
15	GPIO15	R/W	0h	Output Clear bit for this pin
14	GPIO14	R/W	0h	Output Clear bit for this pin
13	GPIO13	R/W	0h	Output Clear bit for this pin
12	GPIO12	R/W	0h	Output Clear bit for this pin
11	GPIO11	R/W	0h	Output Clear bit for this pin
10	GPIO10	R/W	0h	Output Clear bit for this pin

Table 7-164. GPACLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO9	R/W	0h	Output Clear bit for this pin
8	GPIO8	R/W	0h	Output Clear bit for this pin
7	GPIO7	R/W	0h	Output Clear bit for this pin
6	GPIO6	R/W	0h	Output Clear bit for this pin
5	GPIO5	R/W	0h	Output Clear bit for this pin
4	GPIO4	R/W	0h	Output Clear bit for this pin
3	GPIO3	R/W	0h	Output Clear bit for this pin
2	GPIO2	R/W	0h	Output Clear bit for this pin
1	GPIO1	R/W	0h	Output Clear bit for this pin
0	GPIO0	R/W	0h	Output Clear bit for this pin

7.11.5.4 GPATOGGLE Register (Offset = 6h) [reset = 0h]

GPATOGGLE is shown in [Figure 7-153](#) and described in [Table 7-165](#).

GPIO A Data Toggle Register (GPIO0 to 31)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-153. GPATOGGLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-165. GPATOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	Output Toggle Register GPIO pin
30	GPIO30	R/W	0h	Output Toggle Register GPIO pin
29	GPIO29	R/W	0h	Output Toggle Register GPIO pin
28	GPIO28	R/W	0h	Output Toggle Register GPIO pin
27	GPIO27	R/W	0h	Output Toggle Register GPIO pin
26	GPIO26	R/W	0h	Output Toggle Register GPIO pin
25	GPIO25	R/W	0h	Output Toggle Register GPIO pin
24	GPIO24	R/W	0h	Output Toggle Register GPIO pin
23	GPIO23	R/W	0h	Output Toggle Register GPIO pin
22	GPIO22	R/W	0h	Output Toggle Register GPIO pin
21	GPIO21	R/W	0h	Output Toggle Register GPIO pin
20	GPIO20	R/W	0h	Output Toggle Register GPIO pin
19	GPIO19	R/W	0h	Output Toggle Register GPIO pin
18	GPIO18	R/W	0h	Output Toggle Register GPIO pin
17	GPIO17	R/W	0h	Output Toggle Register GPIO pin
16	GPIO16	R/W	0h	Output Toggle Register GPIO pin
15	GPIO15	R/W	0h	Output Toggle Register GPIO pin
14	GPIO14	R/W	0h	Output Toggle Register GPIO pin
13	GPIO13	R/W	0h	Output Toggle Register GPIO pin
12	GPIO12	R/W	0h	Output Toggle Register GPIO pin
11	GPIO11	R/W	0h	Output Toggle Register GPIO pin
10	GPIO10	R/W	0h	Output Toggle Register GPIO pin

Table 7-165. GPATOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO9	R/W	0h	Output Toggle Register GPIO pin
8	GPIO8	R/W	0h	Output Toggle Register GPIO pin
7	GPIO7	R/W	0h	Output Toggle Register GPIO pin
6	GPIO6	R/W	0h	Output Toggle Register GPIO pin
5	GPIO5	R/W	0h	Output Toggle Register GPIO pin
4	GPIO4	R/W	0h	Output Toggle Register GPIO pin
3	GPIO3	R/W	0h	Output Toggle Register GPIO pin
2	GPIO2	R/W	0h	Output Toggle Register GPIO pin
1	GPIO1	R/W	0h	Output Toggle Register GPIO pin
0	GPIO0	R/W	0h	Output Toggle Register GPIO pin

7.11.5.5 GPBDAT Register (Offset = 8h) [reset = 0h]

GPBDAT is shown in [Figure 7-154](#) and described in [Table 7-166](#).

GPIO B Data Register (GPIO32 to 63)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-154. GPBDAT 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-166. GPBDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Data Register for this pin
30	GPIO62	R/W	0h	Data Register for this pin
29	GPIO61	R/W	0h	Data Register for this pin
28	GPIO60	R/W	0h	Data Register for this pin
27	GPIO59	R/W	0h	Data Register for this pin
26	GPIO58	R/W	0h	Data Register for this pin
25	GPIO57	R/W	0h	Data Register for this pin
24	GPIO56	R/W	0h	Data Register for this pin
23	GPIO55	R/W	0h	Data Register for this pin
22	GPIO54	R/W	0h	Data Register for this pin
21	GPIO53	R/W	0h	Data Register for this pin
20	GPIO52	R/W	0h	Data Register for this pin
19	GPIO51	R/W	0h	Data Register for this pin
18	GPIO50	R/W	0h	Data Register for this pin
17	GPIO49	R/W	0h	Data Register for this pin
16	GPIO48	R/W	0h	Data Register for this pin
15	GPIO47	R/W	0h	Data Register for this pin

Table 7-166. GPBDAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	GPIO46	R/W	0h	Data Register for this pin
13	GPIO45	R/W	0h	Data Register for this pin
12	GPIO44	R/W	0h	Data Register for this pin
11	GPIO43	R/W	0h	Data Register for this pin
10	GPIO42	R/W	0h	Data Register for this pin
9	GPIO41	R/W	0h	Data Register for this pin
8	GPIO40	R/W	0h	Data Register for this pin
7	GPIO39	R/W	0h	Data Register for this pin
6	GPIO38	R/W	0h	Data Register for this pin
5	GPIO37	R/W	0h	Data Register for this pin
4	GPIO36	R/W	0h	Data Register for this pin
3	GPIO35	R/W	0h	Data Register for this pin
2	GPIO34	R/W	0h	Data Register for this pin
1	GPIO33	R/W	0h	Data Register for this pin
0	GPIO32	R/W	0h	Data Register for this pin

7.11.5.6 GPBSET Register (Offset = Ah) [reset = 0h]

GPBSET is shown in [Figure 7-155](#) and described in [Table 7-167](#).

GPIO B Data Set Register (GPIO32 to 63)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-155. GPBSET 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-167. GPBSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Output Set bit for this pin
30	GPIO62	R/W	0h	Output Set bit for this pin
29	GPIO61	R/W	0h	Output Set bit for this pin
28	GPIO60	R/W	0h	Output Set bit for this pin
27	GPIO59	R/W	0h	Output Set bit for this pin
26	GPIO58	R/W	0h	Output Set bit for this pin
25	GPIO57	R/W	0h	Output Set bit for this pin
24	GPIO56	R/W	0h	Output Set bit for this pin
23	GPIO55	R/W	0h	Output Set bit for this pin
22	GPIO54	R/W	0h	Output Set bit for this pin
21	GPIO53	R/W	0h	Output Set bit for this pin
20	GPIO52	R/W	0h	Output Set bit for this pin
19	GPIO51	R/W	0h	Output Set bit for this pin
18	GPIO50	R/W	0h	Output Set bit for this pin
17	GPIO49	R/W	0h	Output Set bit for this pin
16	GPIO48	R/W	0h	Output Set bit for this pin
15	GPIO47	R/W	0h	Output Set bit for this pin
14	GPIO46	R/W	0h	Output Set bit for this pin
13	GPIO45	R/W	0h	Output Set bit for this pin
12	GPIO44	R/W	0h	Output Set bit for this pin
11	GPIO43	R/W	0h	Output Set bit for this pin
10	GPIO42	R/W	0h	Output Set bit for this pin

Table 7-167. GPBSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO41	R/W	0h	Output Set bit for this pin
8	GPIO40	R/W	0h	Output Set bit for this pin
7	GPIO39	R/W	0h	Output Set bit for this pin
6	GPIO38	R/W	0h	Output Set bit for this pin
5	GPIO37	R/W	0h	Output Set bit for this pin
4	GPIO36	R/W	0h	Output Set bit for this pin
3	GPIO35	R/W	0h	Output Set bit for this pin
2	GPIO34	R/W	0h	Output Set bit for this pin
1	GPIO33	R/W	0h	Output Set bit for this pin
0	GPIO32	R/W	0h	Output Set bit for this pin

7.11.5.7 GPBCLEAR Register (Offset = Ch) [reset = 0h]

GPBCLEAR is shown in [Figure 7-156](#) and described in [Table 7-168](#).

GPIO B Data Clear Register (GPIO32 to 63)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-156. GPBCLEAR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-168. GPBCLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Output Clear bit for this pin
30	GPIO62	R/W	0h	Output Clear bit for this pin
29	GPIO61	R/W	0h	Output Clear bit for this pin
28	GPIO60	R/W	0h	Output Clear bit for this pin
27	GPIO59	R/W	0h	Output Clear bit for this pin
26	GPIO58	R/W	0h	Output Clear bit for this pin
25	GPIO57	R/W	0h	Output Clear bit for this pin
24	GPIO56	R/W	0h	Output Clear bit for this pin
23	GPIO55	R/W	0h	Output Clear bit for this pin
22	GPIO54	R/W	0h	Output Clear bit for this pin
21	GPIO53	R/W	0h	Output Clear bit for this pin
20	GPIO52	R/W	0h	Output Clear bit for this pin
19	GPIO51	R/W	0h	Output Clear bit for this pin
18	GPIO50	R/W	0h	Output Clear bit for this pin
17	GPIO49	R/W	0h	Output Clear bit for this pin
16	GPIO48	R/W	0h	Output Clear bit for this pin
15	GPIO47	R/W	0h	Output Clear bit for this pin
14	GPIO46	R/W	0h	Output Clear bit for this pin
13	GPIO45	R/W	0h	Output Clear bit for this pin
12	GPIO44	R/W	0h	Output Clear bit for this pin
11	GPIO43	R/W	0h	Output Clear bit for this pin
10	GPIO42	R/W	0h	Output Clear bit for this pin

Table 7-168. GPBCLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO41	R/W	0h	Output Clear bit for this pin
8	GPIO40	R/W	0h	Output Clear bit for this pin
7	GPIO39	R/W	0h	Output Clear bit for this pin
6	GPIO38	R/W	0h	Output Clear bit for this pin
5	GPIO37	R/W	0h	Output Clear bit for this pin
4	GPIO36	R/W	0h	Output Clear bit for this pin
3	GPIO35	R/W	0h	Output Clear bit for this pin
2	GPIO34	R/W	0h	Output Clear bit for this pin
1	GPIO33	R/W	0h	Output Clear bit for this pin
0	GPIO32	R/W	0h	Output Clear bit for this pin

7.11.5.8 GPBTOGGLE Register (Offset = Eh) [reset = 0h]

GPBTOGGLE is shown in [Figure 7-157](#) and described in [Table 7-169](#).

GPIO B Data Toggle Register (GPIO32 to 63)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-157. GPBTOGGLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-169. GPBTOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	Output Toggle Register GPIO pin
30	GPIO62	R/W	0h	Output Toggle Register GPIO pin
29	GPIO61	R/W	0h	Output Toggle Register GPIO pin
28	GPIO60	R/W	0h	Output Toggle Register GPIO pin
27	GPIO59	R/W	0h	Output Toggle Register GPIO pin
26	GPIO58	R/W	0h	Output Toggle Register GPIO pin
25	GPIO57	R/W	0h	Output Toggle Register GPIO pin
24	GPIO56	R/W	0h	Output Toggle Register GPIO pin
23	GPIO55	R/W	0h	Output Toggle Register GPIO pin
22	GPIO54	R/W	0h	Output Toggle Register GPIO pin
21	GPIO53	R/W	0h	Output Toggle Register GPIO pin
20	GPIO52	R/W	0h	Output Toggle Register GPIO pin
19	GPIO51	R/W	0h	Output Toggle Register GPIO pin
18	GPIO50	R/W	0h	Output Toggle Register GPIO pin
17	GPIO49	R/W	0h	Output Toggle Register GPIO pin
16	GPIO48	R/W	0h	Output Toggle Register GPIO pin
15	GPIO47	R/W	0h	Output Toggle Register GPIO pin
14	GPIO46	R/W	0h	Output Toggle Register GPIO pin
13	GPIO45	R/W	0h	Output Toggle Register GPIO pin
12	GPIO44	R/W	0h	Output Toggle Register GPIO pin
11	GPIO43	R/W	0h	Output Toggle Register GPIO pin
10	GPIO42	R/W	0h	Output Toggle Register GPIO pin

Table 7-169. GPBTOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO41	R/W	0h	Output Toggle Register GPIO pin
8	GPIO40	R/W	0h	Output Toggle Register GPIO pin
7	GPIO39	R/W	0h	Output Toggle Register GPIO pin
6	GPIO38	R/W	0h	Output Toggle Register GPIO pin
5	GPIO37	R/W	0h	Output Toggle Register GPIO pin
4	GPIO36	R/W	0h	Output Toggle Register GPIO pin
3	GPIO35	R/W	0h	Output Toggle Register GPIO pin
2	GPIO34	R/W	0h	Output Toggle Register GPIO pin
1	GPIO33	R/W	0h	Output Toggle Register GPIO pin
0	GPIO32	R/W	0h	Output Toggle Register GPIO pin

7.11.5.9 GPCDAT Register (Offset = 10h) [reset = 0h]

GPCDAT is shown in [Figure 7-158](#) and described in [Table 7-170](#).

GPIO C Data Register (GPIO64 to 95)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-158. GPCDAT Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-170. GPCDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Data Register for this pin
30	GPIO94	R/W	0h	Data Register for this pin
29	GPIO93	R/W	0h	Data Register for this pin
28	GPIO92	R/W	0h	Data Register for this pin
27	GPIO91	R/W	0h	Data Register for this pin
26	GPIO90	R/W	0h	Data Register for this pin
25	GPIO89	R/W	0h	Data Register for this pin
24	GPIO88	R/W	0h	Data Register for this pin
23	GPIO87	R/W	0h	Data Register for this pin
22	GPIO86	R/W	0h	Data Register for this pin
21	GPIO85	R/W	0h	Data Register for this pin
20	GPIO84	R/W	0h	Data Register for this pin
19	GPIO83	R/W	0h	Data Register for this pin
18	GPIO82	R/W	0h	Data Register for this pin
17	GPIO81	R/W	0h	Data Register for this pin
16	GPIO80	R/W	0h	Data Register for this pin
15	GPIO79	R/W	0h	Data Register for this pin

Table 7-170. GPCDAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	GPIO78	R/W	0h	Data Register for this pin
13	GPIO77	R/W	0h	Data Register for this pin
12	GPIO76	R/W	0h	Data Register for this pin
11	GPIO75	R/W	0h	Data Register for this pin
10	GPIO74	R/W	0h	Data Register for this pin
9	GPIO73	R/W	0h	Data Register for this pin
8	GPIO72	R/W	0h	Data Register for this pin
7	GPIO71	R/W	0h	Data Register for this pin
6	GPIO70	R/W	0h	Data Register for this pin
5	GPIO69	R/W	0h	Data Register for this pin
4	GPIO68	R/W	0h	Data Register for this pin
3	GPIO67	R/W	0h	Data Register for this pin
2	GPIO66	R/W	0h	Data Register for this pin
1	GPIO65	R/W	0h	Data Register for this pin
0	GPIO64	R/W	0h	Data Register for this pin

7.11.5.10 GPCSET Register (Offset = 12h) [reset = 0h]

GPCSET is shown in [Figure 7-159](#) and described in [Table 7-171](#).

GPIO C Data Set Register (GPIO64 to 95)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-159. GPCSET Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-171. GPCSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Output Set bit for this pin
30	GPIO94	R/W	0h	Output Set bit for this pin
29	GPIO93	R/W	0h	Output Set bit for this pin
28	GPIO92	R/W	0h	Output Set bit for this pin
27	GPIO91	R/W	0h	Output Set bit for this pin
26	GPIO90	R/W	0h	Output Set bit for this pin
25	GPIO89	R/W	0h	Output Set bit for this pin
24	GPIO88	R/W	0h	Output Set bit for this pin
23	GPIO87	R/W	0h	Output Set bit for this pin
22	GPIO86	R/W	0h	Output Set bit for this pin
21	GPIO85	R/W	0h	Output Set bit for this pin
20	GPIO84	R/W	0h	Output Set bit for this pin
19	GPIO83	R/W	0h	Output Set bit for this pin
18	GPIO82	R/W	0h	Output Set bit for this pin
17	GPIO81	R/W	0h	Output Set bit for this pin
16	GPIO80	R/W	0h	Output Set bit for this pin
15	GPIO79	R/W	0h	Output Set bit for this pin
14	GPIO78	R/W	0h	Output Set bit for this pin
13	GPIO77	R/W	0h	Output Set bit for this pin
12	GPIO76	R/W	0h	Output Set bit for this pin
11	GPIO75	R/W	0h	Output Set bit for this pin
10	GPIO74	R/W	0h	Output Set bit for this pin

Table 7-171. GPCSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO73	R/W	0h	Output Set bit for this pin
8	GPIO72	R/W	0h	Output Set bit for this pin
7	GPIO71	R/W	0h	Output Set bit for this pin
6	GPIO70	R/W	0h	Output Set bit for this pin
5	GPIO69	R/W	0h	Output Set bit for this pin
4	GPIO68	R/W	0h	Output Set bit for this pin
3	GPIO67	R/W	0h	Output Set bit for this pin
2	GPIO66	R/W	0h	Output Set bit for this pin
1	GPIO65	R/W	0h	Output Set bit for this pin
0	GPIO64	R/W	0h	Output Set bit for this pin

7.11.5.11 GPCCLEAR Register (Offset = 14h) [reset = 0h]

GPCCLEAR is shown in [Figure 7-160](#) and described in [Table 7-172](#).

GPIO C Data Clear Register (GPIO64 to 95)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-160. GPCCLEAR Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-172. GPCCLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Output Clear bit for this pin
30	GPIO94	R/W	0h	Output Clear bit for this pin
29	GPIO93	R/W	0h	Output Clear bit for this pin
28	GPIO92	R/W	0h	Output Clear bit for this pin
27	GPIO91	R/W	0h	Output Clear bit for this pin
26	GPIO90	R/W	0h	Output Clear bit for this pin
25	GPIO89	R/W	0h	Output Clear bit for this pin
24	GPIO88	R/W	0h	Output Clear bit for this pin
23	GPIO87	R/W	0h	Output Clear bit for this pin
22	GPIO86	R/W	0h	Output Clear bit for this pin
21	GPIO85	R/W	0h	Output Clear bit for this pin
20	GPIO84	R/W	0h	Output Clear bit for this pin
19	GPIO83	R/W	0h	Output Clear bit for this pin
18	GPIO82	R/W	0h	Output Clear bit for this pin
17	GPIO81	R/W	0h	Output Clear bit for this pin
16	GPIO80	R/W	0h	Output Clear bit for this pin
15	GPIO79	R/W	0h	Output Clear bit for this pin
14	GPIO78	R/W	0h	Output Clear bit for this pin
13	GPIO77	R/W	0h	Output Clear bit for this pin
12	GPIO76	R/W	0h	Output Clear bit for this pin
11	GPIO75	R/W	0h	Output Clear bit for this pin
10	GPIO74	R/W	0h	Output Clear bit for this pin

Table 7-172. GPCCLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO73	R/W	0h	Output Clear bit for this pin
8	GPIO72	R/W	0h	Output Clear bit for this pin
7	GPIO71	R/W	0h	Output Clear bit for this pin
6	GPIO70	R/W	0h	Output Clear bit for this pin
5	GPIO69	R/W	0h	Output Clear bit for this pin
4	GPIO68	R/W	0h	Output Clear bit for this pin
3	GPIO67	R/W	0h	Output Clear bit for this pin
2	GPIO66	R/W	0h	Output Clear bit for this pin
1	GPIO65	R/W	0h	Output Clear bit for this pin
0	GPIO64	R/W	0h	Output Clear bit for this pin

7.11.5.12 GPCTOGGLE Register (Offset = 16h) [reset = 0h]

GPCTOGGLE is shown in [Figure 7-161](#) and described in [Table 7-173](#).

GPIO C Data Toggle Register (GPIO64 to 95)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-161. GPCTOGGLE Register

31	30	29	28	27	26	25	24
GPIO95	GPIO94	GPIO93	GPIO92	GPIO91	GPIO90	GPIO89	GPIO88
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
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
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
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
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
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-173. GPCTOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO95	R/W	0h	Output Toggle Register GPIO pin
30	GPIO94	R/W	0h	Output Toggle Register GPIO pin
29	GPIO93	R/W	0h	Output Toggle Register GPIO pin
28	GPIO92	R/W	0h	Output Toggle Register GPIO pin
27	GPIO91	R/W	0h	Output Toggle Register GPIO pin
26	GPIO90	R/W	0h	Output Toggle Register GPIO pin
25	GPIO89	R/W	0h	Output Toggle Register GPIO pin
24	GPIO88	R/W	0h	Output Toggle Register GPIO pin
23	GPIO87	R/W	0h	Output Toggle Register GPIO pin
22	GPIO86	R/W	0h	Output Toggle Register GPIO pin
21	GPIO85	R/W	0h	Output Toggle Register GPIO pin
20	GPIO84	R/W	0h	Output Toggle Register GPIO pin
19	GPIO83	R/W	0h	Output Toggle Register GPIO pin
18	GPIO82	R/W	0h	Output Toggle Register GPIO pin
17	GPIO81	R/W	0h	Output Toggle Register GPIO pin
16	GPIO80	R/W	0h	Output Toggle Register GPIO pin
15	GPIO79	R/W	0h	Output Toggle Register GPIO pin
14	GPIO78	R/W	0h	Output Toggle Register GPIO pin
13	GPIO77	R/W	0h	Output Toggle Register GPIO pin
12	GPIO76	R/W	0h	Output Toggle Register GPIO pin
11	GPIO75	R/W	0h	Output Toggle Register GPIO pin
10	GPIO74	R/W	0h	Output Toggle Register GPIO pin

Table 7-173. GPCTOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO73	R/W	0h	Output Toggle Register GPIO pin
8	GPIO72	R/W	0h	Output Toggle Register GPIO pin
7	GPIO71	R/W	0h	Output Toggle Register GPIO pin
6	GPIO70	R/W	0h	Output Toggle Register GPIO pin
5	GPIO69	R/W	0h	Output Toggle Register GPIO pin
4	GPIO68	R/W	0h	Output Toggle Register GPIO pin
3	GPIO67	R/W	0h	Output Toggle Register GPIO pin
2	GPIO66	R/W	0h	Output Toggle Register GPIO pin
1	GPIO65	R/W	0h	Output Toggle Register GPIO pin
0	GPIO64	R/W	0h	Output Toggle Register GPIO pin

7.11.5.13 GPDDAT Register (Offset = 18h) [reset = 0h]

GPDDAT is shown in [Figure 7-162](#) and described in [Table 7-174](#).

GPIO D Data Register (GPIO96 to 127)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-162. GPDDAT Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-174. GPDDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Data Register for this pin
30	GPIO126	R/W	0h	Data Register for this pin
29	GPIO125	R/W	0h	Data Register for this pin
28	GPIO124	R/W	0h	Data Register for this pin
27	GPIO123	R/W	0h	Data Register for this pin
26	GPIO122	R/W	0h	Data Register for this pin
25	GPIO121	R/W	0h	Data Register for this pin
24	GPIO120	R/W	0h	Data Register for this pin
23	GPIO119	R/W	0h	Data Register for this pin
22	GPIO118	R/W	0h	Data Register for this pin
21	GPIO117	R/W	0h	Data Register for this pin
20	GPIO116	R/W	0h	Data Register for this pin
19	GPIO115	R/W	0h	Data Register for this pin
18	GPIO114	R/W	0h	Data Register for this pin
17	GPIO113	R/W	0h	Data Register for this pin
16	GPIO112	R/W	0h	Data Register for this pin
15	GPIO111	R/W	0h	Data Register for this pin

Table 7-174. GPDDAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	GPIO110	R/W	0h	Data Register for this pin
13	GPIO109	R/W	0h	Data Register for this pin
12	GPIO108	R/W	0h	Data Register for this pin
11	GPIO107	R/W	0h	Data Register for this pin
10	GPIO106	R/W	0h	Data Register for this pin
9	GPIO105	R/W	0h	Data Register for this pin
8	GPIO104	R/W	0h	Data Register for this pin
7	GPIO103	R/W	0h	Data Register for this pin
6	GPIO102	R/W	0h	Data Register for this pin
5	GPIO101	R/W	0h	Data Register for this pin
4	GPIO100	R/W	0h	Data Register for this pin
3	GPIO99	R/W	0h	Data Register for this pin
2	GPIO98	R/W	0h	Data Register for this pin
1	GPIO97	R/W	0h	Data Register for this pin
0	GPIO96	R/W	0h	Data Register for this pin

7.11.5.14 GPDSET Register (Offset = 1Ah) [reset = 0h]

GPDSET is shown in [Figure 7-163](#) and described in [Table 7-175](#).

GPIO D Data Set Register (GPIO96 to 127)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-163. GPDSET Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-175. GPDSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Output Set bit for this pin
30	GPIO126	R/W	0h	Output Set bit for this pin
29	GPIO125	R/W	0h	Output Set bit for this pin
28	GPIO124	R/W	0h	Output Set bit for this pin
27	GPIO123	R/W	0h	Output Set bit for this pin
26	GPIO122	R/W	0h	Output Set bit for this pin
25	GPIO121	R/W	0h	Output Set bit for this pin
24	GPIO120	R/W	0h	Output Set bit for this pin
23	GPIO119	R/W	0h	Output Set bit for this pin
22	GPIO118	R/W	0h	Output Set bit for this pin
21	GPIO117	R/W	0h	Output Set bit for this pin
20	GPIO116	R/W	0h	Output Set bit for this pin
19	GPIO115	R/W	0h	Output Set bit for this pin
18	GPIO114	R/W	0h	Output Set bit for this pin
17	GPIO113	R/W	0h	Output Set bit for this pin
16	GPIO112	R/W	0h	Output Set bit for this pin
15	GPIO111	R/W	0h	Output Set bit for this pin
14	GPIO110	R/W	0h	Output Set bit for this pin
13	GPIO109	R/W	0h	Output Set bit for this pin
12	GPIO108	R/W	0h	Output Set bit for this pin
11	GPIO107	R/W	0h	Output Set bit for this pin
10	GPIO106	R/W	0h	Output Set bit for this pin

Table 7-175. GPDSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO105	R/W	0h	Output Set bit for this pin
8	GPIO104	R/W	0h	Output Set bit for this pin
7	GPIO103	R/W	0h	Output Set bit for this pin
6	GPIO102	R/W	0h	Output Set bit for this pin
5	GPIO101	R/W	0h	Output Set bit for this pin
4	GPIO100	R/W	0h	Output Set bit for this pin
3	GPIO99	R/W	0h	Output Set bit for this pin
2	GPIO98	R/W	0h	Output Set bit for this pin
1	GPIO97	R/W	0h	Output Set bit for this pin
0	GPIO96	R/W	0h	Output Set bit for this pin

7.11.5.15 GPD CLEAR Register (Offset = 1Ch) [reset = 0h]

GPD CLEAR is shown in [Figure 7-164](#) and described in [Table 7-176](#).

GPIO D Data Clear Register (GPIO96 to 127)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-164. GPD CLEAR Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-176. GPD CLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Output Clear bit for this pin
30	GPIO126	R/W	0h	Output Clear bit for this pin
29	GPIO125	R/W	0h	Output Clear bit for this pin
28	GPIO124	R/W	0h	Output Clear bit for this pin
27	GPIO123	R/W	0h	Output Clear bit for this pin
26	GPIO122	R/W	0h	Output Clear bit for this pin
25	GPIO121	R/W	0h	Output Clear bit for this pin
24	GPIO120	R/W	0h	Output Clear bit for this pin
23	GPIO119	R/W	0h	Output Clear bit for this pin
22	GPIO118	R/W	0h	Output Clear bit for this pin
21	GPIO117	R/W	0h	Output Clear bit for this pin
20	GPIO116	R/W	0h	Output Clear bit for this pin
19	GPIO115	R/W	0h	Output Clear bit for this pin
18	GPIO114	R/W	0h	Output Clear bit for this pin
17	GPIO113	R/W	0h	Output Clear bit for this pin
16	GPIO112	R/W	0h	Output Clear bit for this pin
15	GPIO111	R/W	0h	Output Clear bit for this pin
14	GPIO110	R/W	0h	Output Clear bit for this pin
13	GPIO109	R/W	0h	Output Clear bit for this pin
12	GPIO108	R/W	0h	Output Clear bit for this pin
11	GPIO107	R/W	0h	Output Clear bit for this pin
10	GPIO106	R/W	0h	Output Clear bit for this pin

Table 7-176. GPDCLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO105	R/W	0h	Output Clear bit for this pin
8	GPIO104	R/W	0h	Output Clear bit for this pin
7	GPIO103	R/W	0h	Output Clear bit for this pin
6	GPIO102	R/W	0h	Output Clear bit for this pin
5	GPIO101	R/W	0h	Output Clear bit for this pin
4	GPIO100	R/W	0h	Output Clear bit for this pin
3	GPIO99	R/W	0h	Output Clear bit for this pin
2	GPIO98	R/W	0h	Output Clear bit for this pin
1	GPIO97	R/W	0h	Output Clear bit for this pin
0	GPIO96	R/W	0h	Output Clear bit for this pin

7.11.5.16 GPDTOGGLE Register (Offset = 1Eh) [reset = 0h]

GPDTOGGLE is shown in [Figure 7-165](#) and described in [Table 7-177](#).

GPIO D Data Toggle Register (GPIO96 to 127)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-165. GPDTOGGLE Register

31	30	29	28	27	26	25	24
GPIO127	GPIO126	GPIO125	GPIO124	GPIO123	GPIO122	GPIO121	GPIO120
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
GPIO119	GPIO118	GPIO117	GPIO116	GPIO115	GPIO114	GPIO113	GPIO112
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
GPIO111	GPIO110	GPIO109	GPIO108	GPIO107	GPIO106	GPIO105	GPIO104
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
GPIO103	GPIO102	GPIO101	GPIO100	GPIO99	GPIO98	GPIO97	GPIO96
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-177. GPDTOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO127	R/W	0h	Output Toggle Register GPIO pin
30	GPIO126	R/W	0h	Output Toggle Register GPIO pin
29	GPIO125	R/W	0h	Output Toggle Register GPIO pin
28	GPIO124	R/W	0h	Output Toggle Register GPIO pin
27	GPIO123	R/W	0h	Output Toggle Register GPIO pin
26	GPIO122	R/W	0h	Output Toggle Register GPIO pin
25	GPIO121	R/W	0h	Output Toggle Register GPIO pin
24	GPIO120	R/W	0h	Output Toggle Register GPIO pin
23	GPIO119	R/W	0h	Output Toggle Register GPIO pin
22	GPIO118	R/W	0h	Output Toggle Register GPIO pin
21	GPIO117	R/W	0h	Output Toggle Register GPIO pin
20	GPIO116	R/W	0h	Output Toggle Register GPIO pin
19	GPIO115	R/W	0h	Output Toggle Register GPIO pin
18	GPIO114	R/W	0h	Output Toggle Register GPIO pin
17	GPIO113	R/W	0h	Output Toggle Register GPIO pin
16	GPIO112	R/W	0h	Output Toggle Register GPIO pin
15	GPIO111	R/W	0h	Output Toggle Register GPIO pin
14	GPIO110	R/W	0h	Output Toggle Register GPIO pin
13	GPIO109	R/W	0h	Output Toggle Register GPIO pin
12	GPIO108	R/W	0h	Output Toggle Register GPIO pin
11	GPIO107	R/W	0h	Output Toggle Register GPIO pin
10	GPIO106	R/W	0h	Output Toggle Register GPIO pin

Table 7-177. GPDTOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO105	R/W	0h	Output Toggle Register GPIO pin
8	GPIO104	R/W	0h	Output Toggle Register GPIO pin
7	GPIO103	R/W	0h	Output Toggle Register GPIO pin
6	GPIO102	R/W	0h	Output Toggle Register GPIO pin
5	GPIO101	R/W	0h	Output Toggle Register GPIO pin
4	GPIO100	R/W	0h	Output Toggle Register GPIO pin
3	GPIO99	R/W	0h	Output Toggle Register GPIO pin
2	GPIO98	R/W	0h	Output Toggle Register GPIO pin
1	GPIO97	R/W	0h	Output Toggle Register GPIO pin
0	GPIO96	R/W	0h	Output Toggle Register GPIO pin

7.11.5.17 GPEDAT Register (Offset = 20h) [reset = 0h]

GPEDAT is shown in [Figure 7-166](#) and described in [Table 7-178](#).

GPIO E Data Register (GPIO128 to 159)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-166. GPEDAT Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-178. GPEDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Data Register for this pin
30	GPIO158	R/W	0h	Data Register for this pin
29	GPIO157	R/W	0h	Data Register for this pin
28	GPIO156	R/W	0h	Data Register for this pin
27	GPIO155	R/W	0h	Data Register for this pin
26	GPIO154	R/W	0h	Data Register for this pin
25	GPIO153	R/W	0h	Data Register for this pin
24	GPIO152	R/W	0h	Data Register for this pin
23	GPIO151	R/W	0h	Data Register for this pin
22	GPIO150	R/W	0h	Data Register for this pin
21	GPIO149	R/W	0h	Data Register for this pin
20	GPIO148	R/W	0h	Data Register for this pin
19	GPIO147	R/W	0h	Data Register for this pin
18	GPIO146	R/W	0h	Data Register for this pin
17	GPIO145	R/W	0h	Data Register for this pin
16	GPIO144	R/W	0h	Data Register for this pin
15	GPIO143	R/W	0h	Data Register for this pin

Table 7-178. GPEDAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14	GPIO142	R/W	0h	Data Register for this pin
13	GPIO141	R/W	0h	Data Register for this pin
12	GPIO140	R/W	0h	Data Register for this pin
11	GPIO139	R/W	0h	Data Register for this pin
10	GPIO138	R/W	0h	Data Register for this pin
9	GPIO137	R/W	0h	Data Register for this pin
8	GPIO136	R/W	0h	Data Register for this pin
7	GPIO135	R/W	0h	Data Register for this pin
6	GPIO134	R/W	0h	Data Register for this pin
5	GPIO133	R/W	0h	Data Register for this pin
4	GPIO132	R/W	0h	Data Register for this pin
3	GPIO131	R/W	0h	Data Register for this pin
2	GPIO130	R/W	0h	Data Register for this pin
1	GPIO129	R/W	0h	Data Register for this pin
0	GPIO128	R/W	0h	Data Register for this pin

7.11.5.18 GPESET Register (Offset = 22h) [reset = 0h]

GPESET is shown in [Figure 7-167](#) and described in [Table 7-179](#).

GPIO E Data Set Register (GPIO128 to 159)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-167. GPESET Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-179. GPESET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Output Set bit for this pin
30	GPIO158	R/W	0h	Output Set bit for this pin
29	GPIO157	R/W	0h	Output Set bit for this pin
28	GPIO156	R/W	0h	Output Set bit for this pin
27	GPIO155	R/W	0h	Output Set bit for this pin
26	GPIO154	R/W	0h	Output Set bit for this pin
25	GPIO153	R/W	0h	Output Set bit for this pin
24	GPIO152	R/W	0h	Output Set bit for this pin
23	GPIO151	R/W	0h	Output Set bit for this pin
22	GPIO150	R/W	0h	Output Set bit for this pin
21	GPIO149	R/W	0h	Output Set bit for this pin
20	GPIO148	R/W	0h	Output Set bit for this pin
19	GPIO147	R/W	0h	Output Set bit for this pin
18	GPIO146	R/W	0h	Output Set bit for this pin
17	GPIO145	R/W	0h	Output Set bit for this pin
16	GPIO144	R/W	0h	Output Set bit for this pin
15	GPIO143	R/W	0h	Output Set bit for this pin
14	GPIO142	R/W	0h	Output Set bit for this pin
13	GPIO141	R/W	0h	Output Set bit for this pin
12	GPIO140	R/W	0h	Output Set bit for this pin
11	GPIO139	R/W	0h	Output Set bit for this pin
10	GPIO138	R/W	0h	Output Set bit for this pin

Table 7-179. GPESET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO137	R/W	0h	Output Set bit for this pin
8	GPIO136	R/W	0h	Output Set bit for this pin
7	GPIO135	R/W	0h	Output Set bit for this pin
6	GPIO134	R/W	0h	Output Set bit for this pin
5	GPIO133	R/W	0h	Output Set bit for this pin
4	GPIO132	R/W	0h	Output Set bit for this pin
3	GPIO131	R/W	0h	Output Set bit for this pin
2	GPIO130	R/W	0h	Output Set bit for this pin
1	GPIO129	R/W	0h	Output Set bit for this pin
0	GPIO128	R/W	0h	Output Set bit for this pin

7.11.5.19 GPECLEAR Register (Offset = 24h) [reset = 0h]

GPECLEAR is shown in [Figure 7-168](#) and described in [Table 7-180](#).

GPIO E Data Clear Register (GPIO128 to 159)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-168. GPECLEAR Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-180. GPECLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Output Clear bit for this pin
30	GPIO158	R/W	0h	Output Clear bit for this pin
29	GPIO157	R/W	0h	Output Clear bit for this pin
28	GPIO156	R/W	0h	Output Clear bit for this pin
27	GPIO155	R/W	0h	Output Clear bit for this pin
26	GPIO154	R/W	0h	Output Clear bit for this pin
25	GPIO153	R/W	0h	Output Clear bit for this pin
24	GPIO152	R/W	0h	Output Clear bit for this pin
23	GPIO151	R/W	0h	Output Clear bit for this pin
22	GPIO150	R/W	0h	Output Clear bit for this pin
21	GPIO149	R/W	0h	Output Clear bit for this pin
20	GPIO148	R/W	0h	Output Clear bit for this pin
19	GPIO147	R/W	0h	Output Clear bit for this pin
18	GPIO146	R/W	0h	Output Clear bit for this pin
17	GPIO145	R/W	0h	Output Clear bit for this pin
16	GPIO144	R/W	0h	Output Clear bit for this pin
15	GPIO143	R/W	0h	Output Clear bit for this pin
14	GPIO142	R/W	0h	Output Clear bit for this pin
13	GPIO141	R/W	0h	Output Clear bit for this pin
12	GPIO140	R/W	0h	Output Clear bit for this pin
11	GPIO139	R/W	0h	Output Clear bit for this pin
10	GPIO138	R/W	0h	Output Clear bit for this pin

Table 7-180. GPECLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO137	R/W	0h	Output Clear bit for this pin
8	GPIO136	R/W	0h	Output Clear bit for this pin
7	GPIO135	R/W	0h	Output Clear bit for this pin
6	GPIO134	R/W	0h	Output Clear bit for this pin
5	GPIO133	R/W	0h	Output Clear bit for this pin
4	GPIO132	R/W	0h	Output Clear bit for this pin
3	GPIO131	R/W	0h	Output Clear bit for this pin
2	GPIO130	R/W	0h	Output Clear bit for this pin
1	GPIO129	R/W	0h	Output Clear bit for this pin
0	GPIO128	R/W	0h	Output Clear bit for this pin

7.11.5.20 GPETOGGLE Register (Offset = 26h) [reset = 0h]

GPETOGGLE is shown in [Figure 7-169](#) and described in [Table 7-181](#).

GPIO E Data Toggle Register (GPIO128 to 159)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-169. GPETOGGLE Register

31	30	29	28	27	26	25	24
GPIO159	GPIO158	GPIO157	GPIO156	GPIO155	GPIO154	GPIO153	GPIO152
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
GPIO151	GPIO150	GPIO149	GPIO148	GPIO147	GPIO146	GPIO145	GPIO144
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
GPIO143	GPIO142	GPIO141	GPIO140	GPIO139	GPIO138	GPIO137	GPIO136
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
GPIO135	GPIO134	GPIO133	GPIO132	GPIO131	GPIO130	GPIO129	GPIO128
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-181. GPETOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GPIO159	R/W	0h	Output Toggle Register GPIO pin
30	GPIO158	R/W	0h	Output Toggle Register GPIO pin
29	GPIO157	R/W	0h	Output Toggle Register GPIO pin
28	GPIO156	R/W	0h	Output Toggle Register GPIO pin
27	GPIO155	R/W	0h	Output Toggle Register GPIO pin
26	GPIO154	R/W	0h	Output Toggle Register GPIO pin
25	GPIO153	R/W	0h	Output Toggle Register GPIO pin
24	GPIO152	R/W	0h	Output Toggle Register GPIO pin
23	GPIO151	R/W	0h	Output Toggle Register GPIO pin
22	GPIO150	R/W	0h	Output Toggle Register GPIO pin
21	GPIO149	R/W	0h	Output Toggle Register GPIO pin
20	GPIO148	R/W	0h	Output Toggle Register GPIO pin
19	GPIO147	R/W	0h	Output Toggle Register GPIO pin
18	GPIO146	R/W	0h	Output Toggle Register GPIO pin
17	GPIO145	R/W	0h	Output Toggle Register GPIO pin
16	GPIO144	R/W	0h	Output Toggle Register GPIO pin
15	GPIO143	R/W	0h	Output Toggle Register GPIO pin
14	GPIO142	R/W	0h	Output Toggle Register GPIO pin
13	GPIO141	R/W	0h	Output Toggle Register GPIO pin
12	GPIO140	R/W	0h	Output Toggle Register GPIO pin
11	GPIO139	R/W	0h	Output Toggle Register GPIO pin
10	GPIO138	R/W	0h	Output Toggle Register GPIO pin

Table 7-181. GPETOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	GPIO137	R/W	0h	Output Toggle Register GPIO pin
8	GPIO136	R/W	0h	Output Toggle Register GPIO pin
7	GPIO135	R/W	0h	Output Toggle Register GPIO pin
6	GPIO134	R/W	0h	Output Toggle Register GPIO pin
5	GPIO133	R/W	0h	Output Toggle Register GPIO pin
4	GPIO132	R/W	0h	Output Toggle Register GPIO pin
3	GPIO131	R/W	0h	Output Toggle Register GPIO pin
2	GPIO130	R/W	0h	Output Toggle Register GPIO pin
1	GPIO129	R/W	0h	Output Toggle Register GPIO pin
0	GPIO128	R/W	0h	Output Toggle Register GPIO pin

7.11.5.21 GPFDAT Register (Offset = 28h) [reset = 0h]

GPFDAT is shown in [Figure 7-170](#) and described in [Table 7-182](#).

GPIO F Data Register (GPIO160 to 168)

Reading this register reflects the current state of the GPIO pin regardless of which mode the GPIO is in.

Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode. If the GPIO is not in output mode the value written is latched but will not be reflected on the GPIO pin or reads of the GPxDAT register. The written value latched will become active when the GPIO is put into GPIO Output mode

A system reset will clear all bits and latched values to zero.

NOTE: Bit-wise read-modify-write operations should not be performed on this register. For bit-wise operations the GPxSET, GPxCLEAR, or GPxTOGGLE registers should be used instead. If direct writes to GPxDAT are necessary, the entire register should be written at one time.

Figure 7-170. GPFDAT Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-182. GPFDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved

Table 7-182. GPFDAT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Data Register for this pin
7	GPIO167	R/W	0h	Data Register for this pin
6	GPIO166	R/W	0h	Data Register for this pin
5	GPIO165	R/W	0h	Data Register for this pin
4	GPIO164	R/W	0h	Data Register for this pin
3	GPIO163	R/W	0h	Data Register for this pin
2	GPIO162	R/W	0h	Data Register for this pin
1	GPIO161	R/W	0h	Data Register for this pin
0	GPIO160	R/W	0h	Data Register for this pin

7.11.5.22 GPFSET Register (Offset = 2Ah) [reset = 0h]

GPFSET is shown in [Figure 7-171](#) and described in [Table 7-183](#).

GPIO F Data Set Register (GPIO160 to 168)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-171. GPFSET Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-183. GPFSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Output Set bit for this pin
7	GPIO167	R/W	0h	Output Set bit for this pin

Table 7-183. GPFSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	GPIO166	R/W	0h	Output Set bit for this pin
5	GPIO165	R/W	0h	Output Set bit for this pin
4	GPIO164	R/W	0h	Output Set bit for this pin
3	GPIO163	R/W	0h	Output Set bit for this pin
2	GPIO162	R/W	0h	Output Set bit for this pin
1	GPIO161	R/W	0h	Output Set bit for this pin
0	GPIO160	R/W	0h	Output Set bit for this pin

7.11.5.23 GPF CLEAR Register (Offset = 2Ch) [reset = 0h]

GPF CLEAR is shown in [Figure 7-172](#) and described in [Table 7-184](#).

GPIO F Data Clear Register (GPIO160 to 168)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-172. GPF CLEAR Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-184. GPF CLEAR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Output Clear bit for this pin
7	GPIO167	R/W	0h	Output Clear bit for this pin

Table 7-184. GPF CLEAR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	GPIO166	R/W	0h	Output Clear bit for this pin
5	GPIO165	R/W	0h	Output Clear bit for this pin
4	GPIO164	R/W	0h	Output Clear bit for this pin
3	GPIO163	R/W	0h	Output Clear bit for this pin
2	GPIO162	R/W	0h	Output Clear bit for this pin
1	GPIO161	R/W	0h	Output Clear bit for this pin
0	GPIO160	R/W	0h	Output Clear bit for this pin

7.11.5.24 GPFTOGGLE Register (Offset = 2Eh) [reset = 0h]

GPFTOGGLE is shown in [Figure 7-173](#) and described in [Table 7-185](#).

GPIO F Data Toggle Register (GPIO160 to 168)

Writing a 1 will toggle GPIO0 output data latch 1 to 0 or 0 to 1.

Writes of 0 are ignored.

Always reads back a 0.

Figure 7-173. GPFTOGGLE Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO168
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO167	GPIO166	GPIO165	GPIO164	GPIO163	GPIO162	GPIO161	GPIO160
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-185. GPFTOGGLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	GPIO168	R/W	0h	Output Toggle Register GPIO pin
7	GPIO167	R/W	0h	Output Toggle Register GPIO pin

Table 7-185. GPFTOGGLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	GPIO166	R/W	0h	Output Toggle Register GPIO pin
5	GPIO165	R/W	0h	Output Toggle Register GPIO pin
4	GPIO164	R/W	0h	Output Toggle Register GPIO pin
3	GPIO163	R/W	0h	Output Toggle Register GPIO pin
2	GPIO162	R/W	0h	Output Toggle Register GPIO pin
1	GPIO161	R/W	0h	Output Toggle Register GPIO pin
0	GPIO160	R/W	0h	Output Toggle Register GPIO pin

7.11.6 XBAR_REGS Registers

[Table 7-186](#) lists the memory-mapped registers for the XBAR_REGS. All register offset addresses not listed in [Table 7-186](#) should be considered as reserved locations and the register contents should not be modified.

Table 7-186. XBAR_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	XBARFLG1	X-Bar Input Flag Register 1		Go
2h	XBARFLG2	X-Bar Input Flag Register 2		Go
4h	XBARFLG3	X-Bar Input Flag Register 3		Go
8h	XBARCLR1	X-Bar Input Flag Clear Register 1		Go
Ah	XBARCLR2	X-Bar Input Flag Clear Register 2		Go
Ch	XBARCLR3	X-Bar Input Flag Clear Register 3		Go

7.11.6.1 XBARFLG1 Register (Offset = 0h) [reset = 0h]

XBARFLG1 is shown in [Figure 7-174](#) and described in [Table 7-187](#).

X-Bar Input Flag Register 1

Figure 7-174. XBARFLG1 Register

31	30	29	28	27	26	25	24
CMPSS8_CTRL_POUTH	CMPSS8_CTRL_POUTL	CMPSS7_CTRL_POUTH	CMPSS7_CTRL_POUTL	CMPSS6_CTRL_POUTH	CMPSS6_CTRL_POUTL	CMPSS5_CTRL_POUTH	CMPSS5_CTRL_POUTL
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_POUTL	CMPSS3_CTRL_POUTH	CMPSS3_CTRL_POUTL	CMPSS2_CTRL_POUTH	CMPSS2_CTRL_POUTL	CMPSS1_CTRL_POUTH	CMPSS1_CTRL_POUTL
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_PL	CMPSS8_CTRL_PL	CMPSS7_CTRL_PL	CMPSS7_CTRL_PL	CMPSS6_CTRL_PL	CMPSS6_CTRL_PL	CMPSS5_CTRL_PL	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_PL	CMPSS4_CTRL_PL	CMPSS3_CTRL_PL	CMPSS3_CTRL_PL	CMPSS2_CTRL_PL	CMPSS2_CTRL_PL	CMPSS1_CTRL_PL	CMPSS1_CTRL_PL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-187. XBARFLG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CMPSS8_CTRL_POUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
30	CMPSS8_CTRL_POUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
29	CMPSS7_CTRL_POUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
28	CMPSS7_CTRL_POUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-187. XBARFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
27	CMPSS6_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
26	CMPSS6_CTRIPOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
25	CMPSS5_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
24	CMPSS5_CTRIPOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
23	CMPSS4_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
22	CMPSS4_CTRIPOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
21	CMPSS3_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
20	CMPSS3_CTRIPOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-187. XBARFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	CMPSS2_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
18	CMPSS2_CTRIOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
17	CMPSS1_CTRIPOUTH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
16	CMPSS1_CTRIOUTL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
15	CMPSS8_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
14	CMPSS8_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
13	CMPSS7_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
12	CMPSS7_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-187. XBARFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	CMPSS6_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
10	CMPSS6_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
9	CMPSS5_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
8	CMPSS5_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
7	CMPSS4_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
6	CMPSS4_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
5	CMPSS3_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
4	CMPSS3_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-187. XBARFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CMPSS2_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
2	CMPSS2_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
1	CMPSS1_CTRIPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
0	CMPSS1_CTRIPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

7.11.6.2 XBARFLG2 Register (Offset = 2h) [reset = 0h]

XBARFLG2 is shown in [Figure 7-175](#) and described in [Table 7-188](#).

X-Bar Input Flag Register 2

Figure 7-175. XBARFLG2 Register

31	30	29	28	27	26	25	24
ADCCEVT1	ADCB EVT4	ADCB EVT3	ADCB EVT2	ADCB EVT1	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	EXTSYNCOU	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
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
ADCSOCB	ADCSOCA	INPUT7	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-188. XBARFLG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ADCCEVT1	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
30	ADCB EVT4	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
29	ADCB EVT3	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
28	ADCB EVT2	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
27	ADCB EVT1	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-188. XBARFLG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	ADCAEVT4	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
25	ADCAEVT3	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
24	ADCAEVT2	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
23	ADCAEVT1	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
22	EXTSYNCOUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
21	ECAP6_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
20	ECAP5_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
19	ECAP4_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-188. XBARFLG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18	ECAP3_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
17	ECAP2_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
16	ECAP1_OUT	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
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	ADCSOCB	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
6	ADCSOCA	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
5	INPUT7	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
4	INPUT5	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-188. XBARFLG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</p>

7.11.6.3 XBARFLG3 Register (Offset = 4h) [reset = 0h]

XBARFLG3 is shown in [Figure 7-176](#) and described in [Table 7-189](#).

X-Bar Input Flag Register 3

Figure 7-176. XBARFLG3 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED	SD2FLT4_CO MPH	SD2FLT4_CO MPL	SD2FLT3_CO MPH	SD2FLT3_CO MPL	SD2FLT2_CO MPH	SD2FLT2_CO MPL	SD2FLT1_CO MPH
R=0-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
SD2FLT1_CO MPL	SD1FLT4_CO MPH	SD1FLT4_CO MPL	SD1FLT3_CO MPH	SD1FLT3_CO MPL	SD1FLT2_CO MPH	SD1FLT2_CO MPL	SD1FLT1_CO MPH
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_CO MPL	ADCDEVT4	ADCDEVT3	ADCDEVT2	ADCDEVT1	ADCCEVT4	ADCCEVT3	ADCCEVT2
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-189. XBARFLG3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R=0	0h	Reserved
22	SD2FLT4_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
21	SD2FLT4_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
20	SD2FLT3_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
19	SD2FLT3_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-189. XBARFLG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18	SD2FLT2_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
17	SD2FLT2_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
16	SD2FLT1_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
15	SD2FLT1_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
14	SD1FLT4_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
13	SD1FLT4_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
12	SD1FLT3_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
11	SD1FLT3_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-189. XBARFLG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	SD1FLT2_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
9	SD1FLT2_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
8	SD1FLT1_COMPH	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
7	SD1FLT1_COMPL	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
6	ADCDEVT4	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
5	ADCDEVT3	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
4	ADCDEVT2	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software
3	ADCDEVT1	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: Corresponding Input was triggered 0: Corresponding Input was not triggered Note: [1] setting of this bit has priority over clear by software

Table 7-189. XBARFLG3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</p>
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: Corresponding Input was triggered</p> <p>0: Corresponding Input was not triggered</p> <p>Note:</p> <p>[1] setting of this bit has priority over clear by software</p>

7.11.6.4 XBARCLR1 Register (Offset = 8h) [reset = 0h]

XBARCLR1 is shown in [Figure 7-177](#) and described in [Table 7-190](#).

X-Bar Input Flag Clear Register 1

Figure 7-177. 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/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
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/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
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/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
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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-190. XBARCLR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CMPSS8_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
30	CMPSS8_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
29	CMPSS7_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
28	CMPSS7_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
27	CMPSS6_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
26	CMPSS6_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
25	CMPSS5_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
24	CMPSS5_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
23	CMPSS4_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect

Table 7-190. XBARCLR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
22	CMPSS4_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
21	CMPSS3_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
20	CMPSS3_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
19	CMPSS2_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
18	CMPSS2_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
17	CMPSS1_CTRIPOUTH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
16	CMPSS1_CTRIPOUTL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
15	CMPSS8_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
14	CMPSS8_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
13	CMPSS7_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
12	CMPSS7_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
11	CMPSS6_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
10	CMPSS6_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
9	CMPSS5_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
8	CMPSS5_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
7	CMPSS4_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
6	CMPSS4_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect

Table 7-190. XBARCLR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	CMPSS3_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
4	CMPSS3_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
3	CMPSS2_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
2	CMPSS2_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
1	CMPSS1_CTRIPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect
0	CMPSS1_CTRIPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect

7.11.6.5 XBARCLR2 Register (Offset = Ah) [reset = 0h]

XBARCLR2 is shown in [Figure 7-178](#) and described in [Table 7-191](#).

X-Bar Input Flag Clear Register 2

Figure 7-178. XBARCLR2 Register

31	30	29	28	27	26	25	24
ADCCEVT1	ADCB EVT4	ADCB EVT3	ADCB EVT2	ADCB EVT1	ADCAEVT4	ADCAEVT3	ADCAEVT2
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
23	22	21	20	19	18	17	16
ADCAEVT1	EXTSYNCOU	ECAP6_OUT	ECAP5_OUT	ECAP4_OUT	ECAP3_OUT	ECAP2_OUT	ECAP1_OUT
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
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
ADCSOCB	ADCSOCA	INPUT7	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-191. XBARCLR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ADCCEVT1	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
30	ADCB EVT4	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
29	ADCB EVT3	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
28	ADCB EVT2	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
27	ADCB EVT1	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
26	ADCAEVT4	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
25	ADCAEVT3	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
24	ADCAEVT2	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
23	ADCAEVT1	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
22	EXTSYNCOU	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect

Table 7-191. XBARCLR2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21	ECAP6_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
20	ECAP5_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
19	ECAP4_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
18	ECAP3_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
17	ECAP2_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
16	ECAP1_OUT	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
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	ADCSOCB	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
6	ADCSOCA	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
5	INPUT7	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
4	INPUT5	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
3	INPUT4	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
2	INPUT3	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
1	INPUT2	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect
0	INPUT1	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG2 register. Writing 0 has no effect

7.11.6.6 XBARCLR3 Register (Offset = Ch) [reset = 0h]

XBARCLR3 is shown in [Figure 7-179](#) and described in [Table 7-192](#).

X-Bar Input Flag Clear Register 3

Figure 7-179. XBARCLR3 Register

31	30	29	28	27	26	25	24
RESERVED							
R=0-0h							
23	22	21	20	19	18	17	16
RESERVED	SD2FLT4_CO MPH	SD2FLT4_CO MPL	SD2FLT3_CO MPH	SD2FLT3_CO MPL	SD2FLT2_CO MPH	SD2FLT2_CO MPL	SD2FLT1_CO MPH
R=0-0h	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
SD2FLT1_CO MPL	SD1FLT4_CO MPH	SD1FLT4_CO MPL	SD1FLT3_CO MPH	SD1FLT3_CO MPL	SD1FLT2_CO MPH	SD1FLT2_CO MPL	SD1FLT1_CO MPH
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
SD1FLT1_CO MPL	ADCDEVT4	ADCDEVT3	ADCDEVT2	ADCDEVT1	ADCCEVT4	ADCCEVT3	ADCCEVT2
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 7-192. XBARCLR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R=0	0h	Reserved
22	SD2FLT4_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
21	SD2FLT4_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
20	SD2FLT3_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
19	SD2FLT3_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
18	SD2FLT2_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
17	SD2FLT2_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
16	SD2FLT1_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
15	SD2FLT1_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
14	SD1FLT4_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect

Table 7-192. XBARCLR3 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
13	SD1FLT4_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
12	SD1FLT3_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
11	SD1FLT3_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
10	SD1FLT2_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
9	SD1FLT2_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
8	SD1FLT1_COMPH	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
7	SD1FLT1_COMPL	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
6	ADCDEVT4	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
5	ADCDEVT3	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
4	ADCDEVT2	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
3	ADCDEVT1	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
2	ADCCEVT4	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
1	ADCCEVT3	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect
0	ADCCEVT2	R=0/W=1	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG3 register. Writing 0 has no effect

Analog Subsystem

This analog subsystem module is described in this chapter.

Topic	Page
8.1 Analog Subsystem	1179
8.2 Registers	1183

8.1 Analog Subsystem

The analog modules on this device include the Analog-to-Digital Converter (ADC), Temperature Sensor, Buffered Digital-to-Analog Converter (DAC), and Comparator Subsystem (CMPSS).

8.1.1 Features

The analog subsystem has the following features:

- Flexible voltage references
 - VREFHIA and VREFLOA, VREFHIB and VREFLOB, VREFHIC and VREFLOC, and VREFHID and VREFLOD externally supplied reference voltage pins
 - Selectable by ADCs and buffered DACs
 - VDAC externally supplied reference voltage pin
 - Selectable by buffered DACs and comparator subsystem DACs
 - Low reference is VSSA
- Flexible pin usage
 - Buffered DAC and comparator subsystem functions multiplexed with ADC inputs
- Internal connection to VREFLO on all ADCs for offset self-calibration

8.1.2 Block Diagram

The subsystem block diagrams show the connections between the different integrated analog modules and to the device pins. These pins fall into two categories: analog module inputs/outputs and reference pins.

The reference pins, VREFHIA to VREFHID and VREFLOA to VREFLOD, can be used to externally supply the reference to the ADC. VREFHIA can also be used to supply the reference voltage to DAC A and DAC B and VREFHIB can be used to supply the reference to DAC C.

The analog module input/outputs are all ADC inputs by default. The pins which connect to CMPSS inputs can be used for the CMPSS without further action and without preventing use as an ADC input simultaneously. DAC outputs must be enabled; this will prevent the channel from simultaneously being used as an ADC input (but the ADC can be used to sample the DAC output voltage if desired).

The VDAC reference pin can be used to set an alternate range for DAC A, DAC B, and DAC C and for the DACs inside the CMPSS modules (the CMPSS DACs are referenced to VDDA and VSSA by default). Using this pin as a reference will prevent the channel from being used as an ADC input (but the ADC can be used to sample the VDAC voltage if desired). The choice of reference is configurable per-module for each CMPSS or buffered DAC and the selection is made via the module's configuration registers.

The block diagram for the analog subsystem is presented in the following graphics.

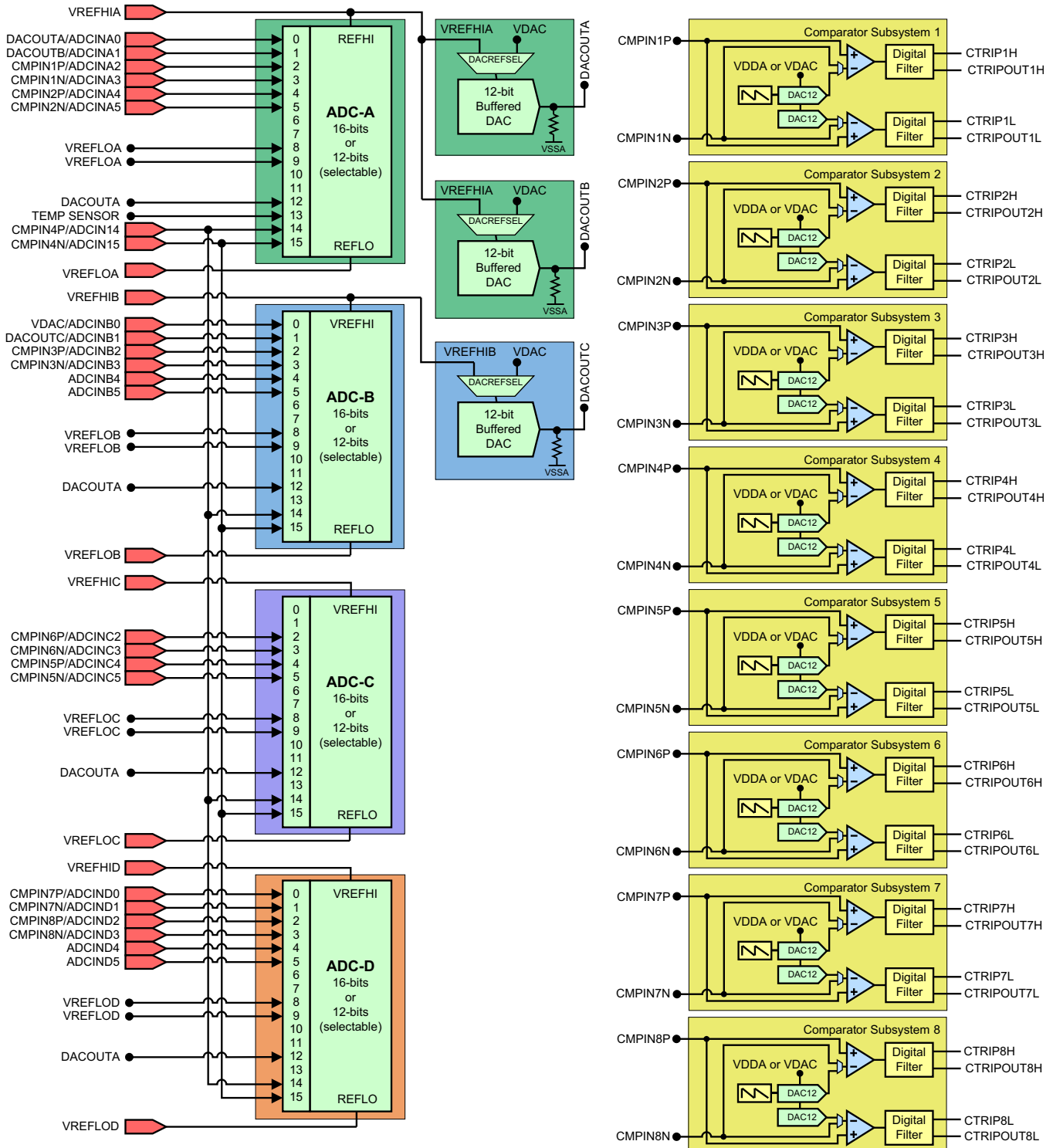
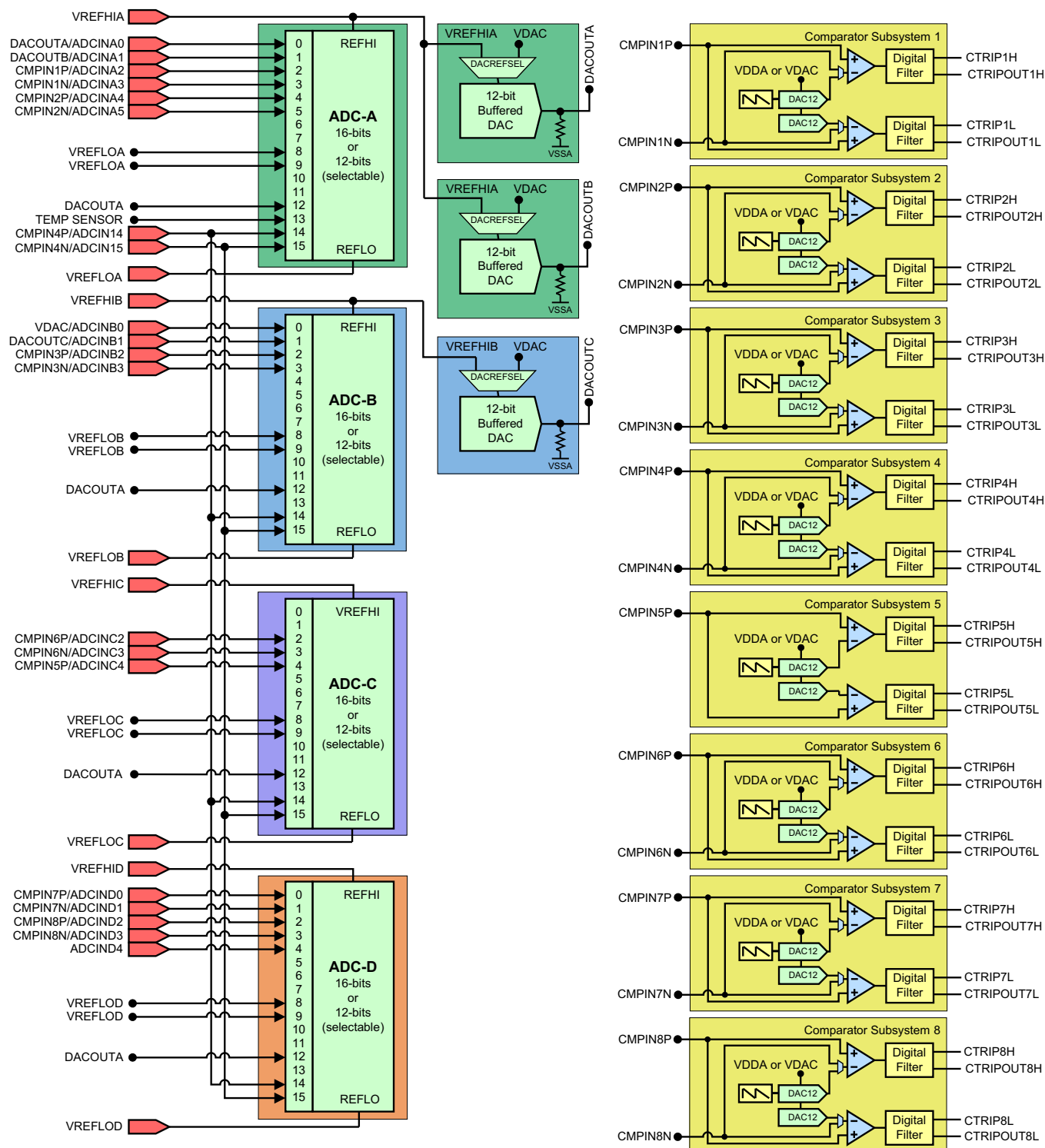
Figure 8-1. Analog Subsystem Block Diagram (337-Ball ZWT)


Figure 8-2. Analog Subsystem Block Diagram (176-Pin PTP)



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 VREFHIx 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.

8.1.3 Lock Registers

Setting the TSNSCTL bit in the LOCK register will disable any further changes in the TSNSCTL register.

8.2 Registers

8.2.1 Analog Subsystem Base Addresses

Table 8-1. Analog Subsystem Base Address Table

AnalogSubsysRegs	ANALOG_SUBSYS_REGS	0x0005_D180	0x0005_D1FF
------------------	--------------------	-------------	-------------

8.2.2 ANALOG_SUBSYS_REGS Registers

Table 8-2 lists the memory-mapped registers for the ANALOG_SUBSYS_REGS. All register offset addresses not listed in Table 8-2 should be considered as reserved locations and the register contents should not be modified.

Table 8-2. ANALOG_SUBSYS_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
20h	INTOSC1TRIM	Internal Oscillator 1 Trim Register	EALLOW	Go
22h	INTOSC2TRIM	Internal Oscillator 2 Trim Register	EALLOW	Go
26h	TSNSCTL	Temperature Sensor Control Register	EALLOW	Go
2Eh	LOCK	Lock Register	EALLOW	Go
36h	ANAREFTRIMA	Analog Reference Trim A Register	EALLOW	Go
38h	ANAREFTRIMB	Analog Reference Trim B Register	EALLOW	Go
3Ah	ANAREFTRIMC	Analog Reference Trim C Register	EALLOW	Go
3Ch	ANAREFTRIMD	Analog Reference Trim D Register	EALLOW	Go

8.2.2.1 INTOSC1TRIM Register (Offset = 20h) [reset = 0h]

INTOSC1TRIM is shown in [Figure 8-3](#) and described in [Table 8-3](#).

Internal Oscillator 1 Trim Register

Figure 8-3. INTOSC1TRIM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-3. INTOSC1TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-0	VALFINETRIM	R/W	0h	Oscillator Value Fine Trim Bits. 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.

8.2.2.2 INTOSC2TRIM Register (Offset = 22h) [reset = 0h]

INTOSC2TRIM is shown in [Figure 8-4](#) and described in [Table 8-4](#).

Internal Oscillator 2 Trim Register

Figure 8-4. INTOSC2TRIM Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-4. INTOSC2TRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-0	VALFINETRIM	R/W	0h	Oscillator Value Fine Trim Bits. 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.

8.2.2.3 TSNSCTL Register (Offset = 26h) [reset = 0h]

TSNSCTL is shown in [Figure 8-5](#) and described in [Table 8-5](#).

Temperature Sensor Control Register

Figure 8-5. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-5. 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

8.2.2.4 LOCK Register (Offset = 2Eh) [reset = 0h]

LOCK is shown in [Figure 8-6](#) and described in [Table 8-6](#).

Lock Register

Figure 8-6. LOCK Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED		
R-0h	R-0h	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	RESERVED	RESERVED	RESERVED	TSNSCTL	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R/WOnce-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-6. LOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
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	RESERVED	R	0h	Reserved
22	RESERVED	R	0h	Reserved
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
18-7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	TSNSCTL	R/WOnce	0h	Temperature Sensor Control Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

8.2.2.5 ANAREFTRIMA Register (Offset = 36h) [reset = 0h]

ANAREFTRIMA is shown in [Figure 8-7](#) and described in [Table 8-7](#).

Analog Reference Trim A Register

Figure 8-7. ANAREFTRIMA Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-7. ANAREFTRIMA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-11	IREFTRIM	R/W	0h	Reference Current Trim. This bit field defines the reference current trim value. 0x0 - Untrimmed all other values - Trimmed
10-6	BGSLOPETRIM	R/W	0h	Bandgap Slope Trim. This bit field defines the bandgap slope trim value. 0x0 - Untrimmed all other values - Trimmed
5-0	BGVALTRIM	R/W	0h	Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value. 0x0 - Untrimmed all other values - Trimmed

8.2.2.6 ANAREFTRIMB Register (Offset = 38h) [reset = 0h]

ANAREFTRIMB is shown in [Figure 8-8](#) and described in [Table 8-8](#).

Analog Reference Trim B Register

Figure 8-8. ANAREFTRIMB Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-8. ANAREFTRIMB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-11	IREFTRIM	R/W	0h	Reference Current Trim. This bit field defines the reference current trim value. 0x0 - Untrimmed all other values - Trimmed
10-6	BGSLOPETRIM	R/W	0h	Bandgap Slope Trim. This bit field defines the bandgap slope trim value. 0x0 - Untrimmed all other values - Trimmed
5-0	BGVALTRIM	R/W	0h	Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value. 0x0 - Untrimmed all other values - Trimmed

8.2.2.7 ANAREFTRIMC Register (Offset = 3Ah) [reset = 0h]

ANAREFTRIMC is shown in [Figure 8-9](#) and described in [Table 8-9](#).

Analog Reference Trim C Register

Figure 8-9. ANAREFTRIMC Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-9. ANAREFTRIMC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-11	IREFTRIM	R/W	0h	Reference Current Trim. This bit field defines the reference current trim value. 0x0 - Untrimmed all other values - Trimmed
10-6	BGSLOPETRIM	R/W	0h	Bandgap Slope Trim. This bit field defines the bandgap slope trim value. 0x0 - Untrimmed all other values - Trimmed
5-0	BGVALTRIM	R/W	0h	Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value. 0x0 - Untrimmed all other values - Trimmed

8.2.2.8 ANAREFTRIMD Register (Offset = 3Ch) [reset = 0h]

ANAREFTRIMD is shown in [Figure 8-10](#) and described in [Table 8-10](#).

Analog Reference Trim D Register

Figure 8-10. ANAREFTRIMD Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 8-10. ANAREFTRIMD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	0h	Reserved
15-11	IREFTRIM	R/W	0h	Reference Current Trim. This bit field defines the reference current trim value. 0x0 - Untrimmed all other values - Trimmed
10-6	BGSLOPETRIM	R/W	0h	Bandgap Slope Trim. This bit field defines the bandgap slope trim value. 0x0 - Untrimmed all other values - Trimmed
5-0	BGVALTRIM	R/W	0h	Bandgap Value Trim. This bit field defines the bandgap voltage offset trim value. 0x0 - Untrimmed all other values - Trimmed

Analog-to-Digital Converter (ADC)

The analog-to-digital converter module described in this chapter is a Type 4 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
9.1 Analog-to-Digital Converter (ADC)	1194
9.2 ADC Timings	1213
9.3 Additional Information	1218
9.4 Registers	1221

9.1 Analog-to-Digital Converter (ADC)

The ADC module described here is a successive approximation (SAR) style ADC with selectable resolution of either 16 bits or 12 bits. This chapter refers to the analog circuits of the converter as the “core,” and includes the channel select MUX, the sample-and-hold (S/H) circuit, the successive approximation circuits, voltage reference circuits, and other analog support circuits. The digital circuits of the converter are referred to as the “wrapper” and includes 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.

9.1.1 Features

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 9.1.4](#)).

Each ADC has the following features:

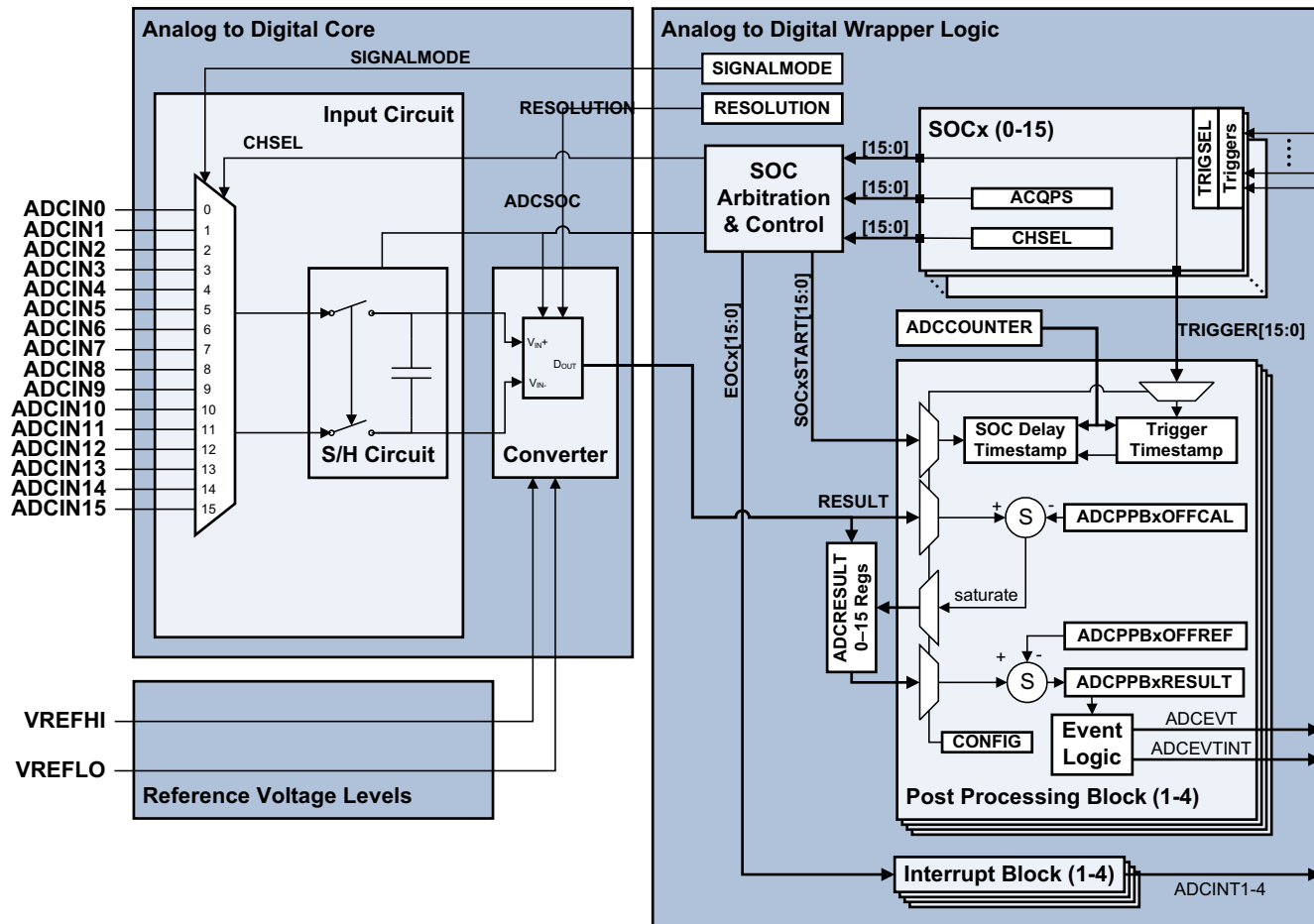
- Selectable resolution of 12 bits or 16 bits
- Ratiometric external reference set by VREFHI\VREFLO
- Differential signal conversions (16-bit mode only)
- Single-ended signal conversions (12-bit mode only)
- Input multiplexer with up to 16 channels (single-ended) or 8 channels (differential)
- 16 configurable SOC
- 16 individually addressable result registers
- Multiple trigger sources
 - S/W - software immediate start
 - All ePWMs - ADCSOC A,B,C, or D
 - GPIO XINT2
 - CPU Timers 0/1/2 (from each C28x core present)
 - ADCINT1/2
- Four flexible PIE interrupts
- 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.

9.1.2 ADC Block Diagram

The block diagram for the ADC core and ADC wrapper are presented in [Figure 9-1](#).

Figure 9-1. ADC Module Block Diagram



9.1.3 ADC Configurability

Some ADC configurations are individually controlled by the SOCx, while others are globally controlled per ADC module. Table 9-1 summarizes the basic ADC options and their level of configurability. The subsequent sections discuss these configurations.

Table 9-1. ADC Options and Configuration Levels

Options	Configurability
Clock	Per module
Resolution (16-bits or 12-bits)	Per module
Signal mode (differential or single-ended)	Per module
Trigger source	Per SOC
Converted channel	Per SOC
Acquisition window duration	Per SOC
EOC location	Per module
Burst Mode	Per module

9.1.3.1 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.

In 16-bit mode, the core requires approximately 29.5 ADCCLK cycles to process a voltage into a conversion result, while in 12-bit mode, this process requires approximately 10.5 ADCCLK cycles. The choice of resolution will also determine the necessary duration of the acquisition window, see [Section 9.3.1](#).

NOTE: To determine an appropriate value for ADCCTL2.PRESCALE, consult the datasheet of your device to determine the maximum SYSCLK and ADCCLK frequency.

9.1.3.2 Resolution

The resolution of the ADC determines how finely the analog range is quantized into digital values. This ADC supports a configurable resolution of 16 bits or 12 bits. The resolution should be configured by using the `AdcSetMode` function provided in `ControlSUITE` in `F2837xD_Adc.c`. This function ensures that the correct trim is loaded into the ADC trim registers. This function must be called at least once after a device reset. The resolution should not be configured by writing to the ADCCTL2 register directly. The resolution can be changed at any time when the ADC is idle (no active or pending SOC's). No wait time is necessary after changing the resolution before conversions can be initiated. If SOC's are active or pending when the resolution is changed, those SOC's may produce incorrect conversion results.

9.1.3.3 Voltage Reference

Each ADC has a VREFHI input and a VREFLO input, which is used as a ratiometric reference.

See [Section 9.3.3](#) 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.

9.1.3.4 Signal Mode

The ADC supports two signal modes: single-ended and differential. In single-ended mode, the input voltage to the converter is sampled through a single pin (ADCINx), referenced to VREFLO. In differential signaling mode, the input voltage to the converter is sampled through a pair of input pins, one of which is the positive input (ADCINxP) and the other is the negative input (ADCINxN). The actual input voltage is the difference between the two (ADCINxP – ADCINxN).

The signal mode should be configured by using the `AdcSetMode` function provided in `ControlSUITE` in `F2837xD_Adc.c`. This function ensures that the correct trim is loaded into the ADC trim registers. This function must be called at least once after a device reset. The signal mode should not be configured by writing to the ADCCTL2 register directly.

NOTES:

- In 16-bit differential signaling mode, VREFLO must be connected to VSSA.
- In differential signal mode, the common mode voltage is

$$V_{CM} = (ADCINxP + ADCINxN)/2$$

The datasheet for a particular device will place some requirements on how close this voltage needs to be to

$$(VREFHI + VREFLO)/2$$

Note: The above condition is not met by connecting the negative input to VSSA or VREFLO.

- Differential signaling mode is advantageous because noise encountered on both inputs will be largely cancelled. The effect can be maximized by routing the positive and negative traces for the same differential input as close together as possible and keeping them symmetrical with respect to the signal reference.

9.1.3.5 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 9-2. Analog to 12-bit Digital Formulas

	Analog Input	Digital Result
Single-Ended	when $ADCINy \leq VREFLO$	$ADCRESULTx = 0$
	when $VREFLO < ADCINy < VREFHI$	$ADCRESULTx = 4096 \left(\frac{ADCINy - VREFLO}{VREFHI - VREFLO} \right)$
	when $ADCINy \geq VREFHI$	$ADCRESULTx = 4095$
Differential	Invalid Mode	Invalid Mode

Table 9-3. Analog to 16-bit Digital Formulas

	Analog Input	Digital Result
Single-Ended	Invalid Mode	Invalid Mode
Differential	when $ADCINyP - ADCINyN \geq VREFHI$	$ADCRESULTx = 0$
	when $-VREFHI < ADCINyP - ADCINyN < VREFHI$	$ADCRESULTx = 65536 \left(\frac{ADCINyP - ADCINyN + VREFHI}{2 VREFHI} \right)$
	when $ADCINyP - ADCINyN \geq VREFHI$	$ADCRESULTx = 65535$

9.1.3.6 Interpreting Conversion Results

Based on a given ADC conversion result, the ideal corresponding analog input is given by the below tables. This corresponds to the center of the possible range of analog voltages that could have produced this conversion result.

Table 9-4. 12-Bit Digital-to-Analog Formulas

	Digital Value	Analog Equivalent
Single-Ended	when $ADCRESULTy = 0$	$ADCINx \leq VREFLO$
	when $0 < ADCRESULTy < 4095$	$ADCINx = (VREFHI - VREFLO) \left(\frac{ADCRESULTy}{4096} \right) + VREFLO$
	when $ADCRESULTy = 4095$	$ADCINx \geq VREFHI$
Differential	Invalid Mode	Invalid Mode

Table 9-5. 16-Bit Digital-to-Analog Formulas

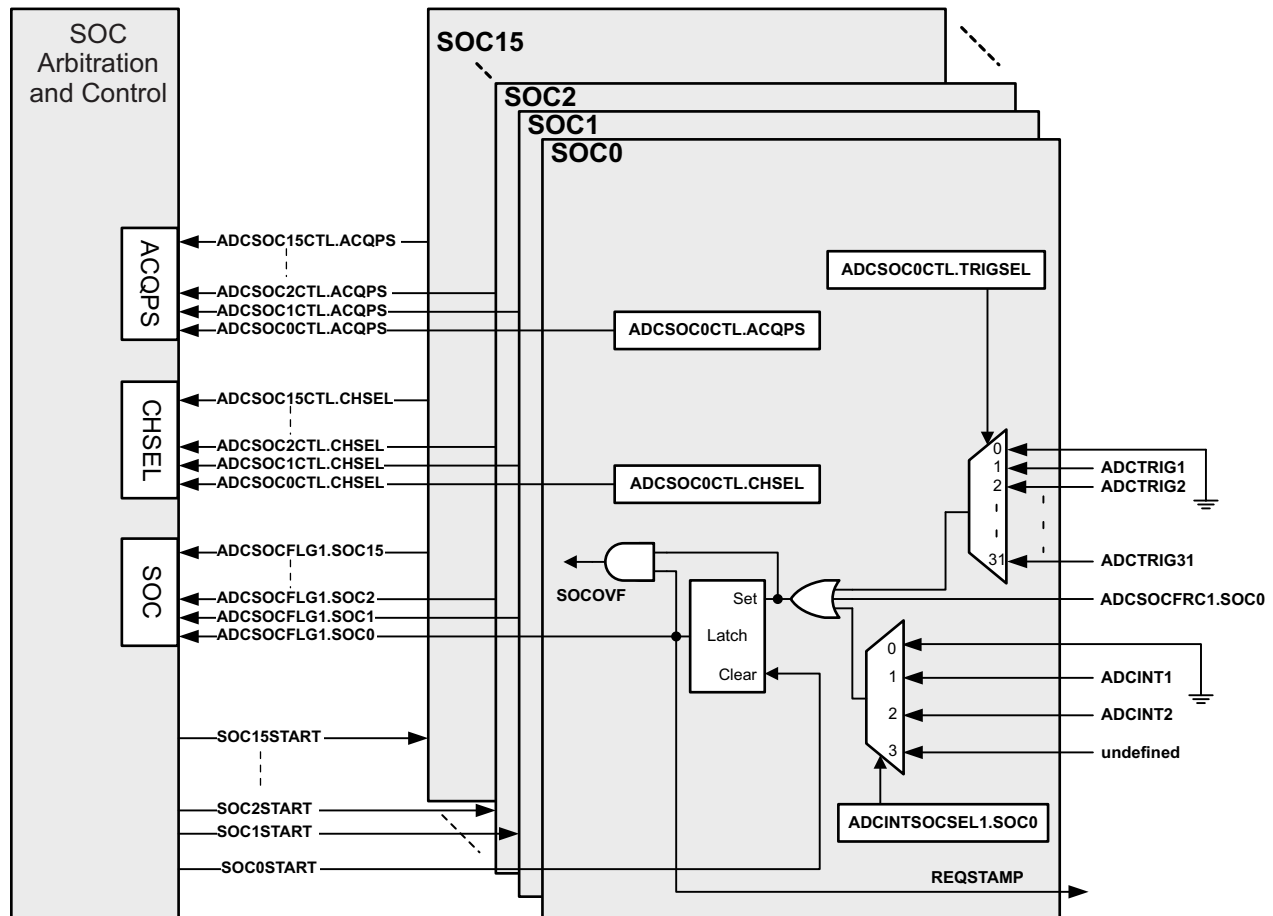
	Digital Value	Analog Equivalent
Single-Ended	Invalid Mode	Invalid Mode
Differential	when $ADCRESULTy = 0$	$ADCINxP - ADCINxN \leq -VREFHI$
	when $0 < ADCRESULTy < 65535$	$ADCINxP - ADCINxN = VREFHI \left(\frac{2 ADCRESULTy}{65536} - 1 \right)$
	when $ADCRESULTy = 65535$	$ADCINxP - ADCINxN \geq VREFHI$

9.1.4 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 9-2. SOC Block Diagram



9.1.4.1 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.

9.1.4.2 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 (from each C28x core present)
- GPIO: Input X-Bar INPUT5
- ePWM1 to ePWM12, ADCSOCA/C or ADCSOCB/D

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.

9.1.4.3 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:

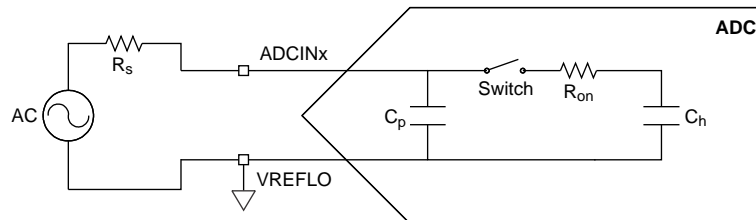
Acquisition window = (ACQPS + 1) · (System Clock (SYSCLK) cycle time)

- 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.

9.1.4.4 ADC Input Models

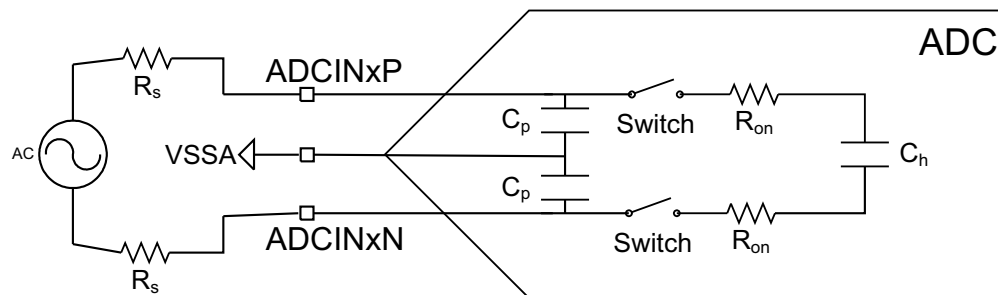
For single-ended operation, the ADC input characteristics for values in the single-ended input model (see [Figure 9-3](#)) can be found in the device data manual.

Figure 9-3. Single-Ended Input Model



For differential operation, the ADC input characteristics for values in the differential input model (see [Figure 9-4](#)) can be found in the device data manual.

Figure 9-4. Differential Input Model



These input models should be used along with actual signal source impedance to determine the acquisition window duration. See [Section 9.3.1](#) for more information.

9.1.4.5 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. Depending on the signal mode, the selection is different. For single-ended signal mode, the value in CHSEL selects a single pin as the input. For differential signal mode, the value in CHSEL selects an even-odd pin pair to be the positive and negative inputs. This is summarized in [Table 9-6](#).

Table 9-6. 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	
	CHSEL	Positive Input	Negative Input
Differential	0 or 1	ADCIN0	ADCIN1
	2 or 3	ADCIN2	ADCIN3
	4 or 5	ADCIN4	ADCIN5
	6 or 7	ADCIN6	ADCIN7
	8 or 9	ADCIN8	ADCIN9
	10 or 11	ADCIN10	ADCIN11
	12 or 13	ADCIN12	ADCIN13
	14 or 15	ADCIN14	ADCIN15

9.1.5 SOC Configuration Examples

The following sections provide some specific examples of how to configure the SOC5s to produce some conversions.

9.1.5.1 Single Conversion from ePWM Trigger

To configure ADC5 to perform a single conversion on channel ADCIN1 when the ePWM timer reaches its period match, a few things are necessary. First, ePWM3 must be configured to generate an SOCA, SOCB, SOCC, or SOCD 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 SOC5s could be used.

Assuming a 100ns sample window is desired with a SYSCLK frequency of 200MHz, then the acquisition window duration should be $100\text{ns}/5\text{ns} = 20$ SYSCLK cycles. The ACQPS field should therefore be set to $20 - 1 = 19$.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 will convert ADCIN1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 19;     //SOC5 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;   //SOC5 will begin conversion on ePWM3 SOCB/D
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCIN1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCIN1 will begin sampling when SOC5 gains priority (see [Section 9.1.6](#)). The ADC control logic will sample ADCIN1 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 9.2](#) for exact sample, conversion, and result latch timings).

9.1.5.2 Oversampled Conversion from ePWM Trigger

To configure the ADC to oversample ADCIN1 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 ADCIN1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 19;     //SOC5 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;   //SOC5 will begin conversion on ePWM3 SOCB/D
AdcaRegs.ADCSOC6CTL.bit.CHSEL = 1;      //SOC6 will convert ADCIN1
AdcaRegs.ADCSOC6CTL.bit.ACQPS = 19;     //SOC6 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC6CTL.bit.TRIGSEL = 10;   //SOC6 will begin conversion on ePWM3 SOCB/D
AdcaRegs.ADCSOC7CTL.bit.CHSEL = 1;      //SOC7 will convert ADCIN1
AdcaRegs.ADCSOC7CTL.bit.ACQPS = 19;     //SOC7 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC7CTL.bit.TRIGSEL = 10;   //SOC7 will begin conversion on ePWM3 SOCB/D
AdcaRegs.ADCSOC8CTL.bit.CHSEL = 1;      //SOC8 will convert ADCIN1
AdcaRegs.ADCSOC8CTL.bit.ACQPS = 19;     //SOC8 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC8CTL.bit.TRIGSEL = 10;   //SOC8 will begin conversion on ePWM3 SOCB/D
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCIN1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCIN1 will begin sampling when SOC5 gains priority (see [ADC Conversion Priority](#)). Once the conversion is complete for SOC5, SOC6 will begin converting ADCIN1 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.

9.1.5.3 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 SYSCLK cycles for each signal, then the ACQPS register setting. This is shown in [Table 9-7](#), where a SYCLK of 200MHz is assumed (5ns cycle time).

Table 9-7. Example Requirements for Multiple Signal Sampling

Signal Name	Acquisition Window Requirement (ns)	Acquisition Window SYSCLK Cycles	ACQPS Register Value
Signal 1	>120ns	120ns/5ns = 24	24 – 1 = 23
Signal 2	>444ns	444ns/5ns = 89 (round up)	89 – 1 = 88
Signal 3	>110ns	110ns/5ns = 22	22 – 1 = 21
Signal 4	>291ns	291ns/5ns = 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 9-8](#)).

Table 9-8. 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 SYSCLK 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 SYSCLK 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 SYSCLK 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 SYSCLK 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.

9.1.5.4 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
```

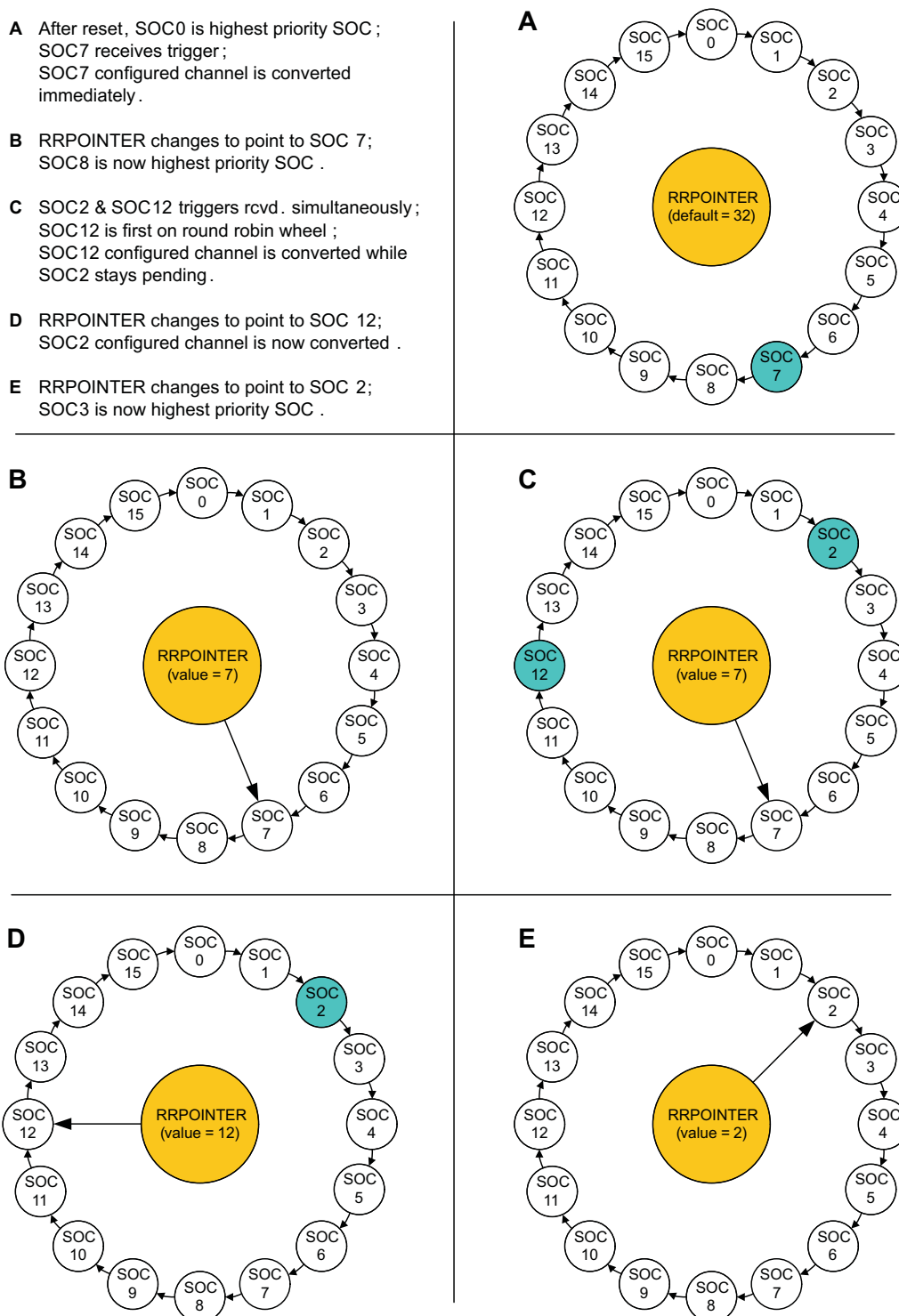
9.1.6 ADC Conversion Priority

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 32 since 0 indicates a conversion has already occurred. When RRPOINTER equals 32 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 SOCPRICCTL register is written.

An example of the round robin priority method is given in [Figure 9-5](#).

Figure 9-5. 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 9-6](#).

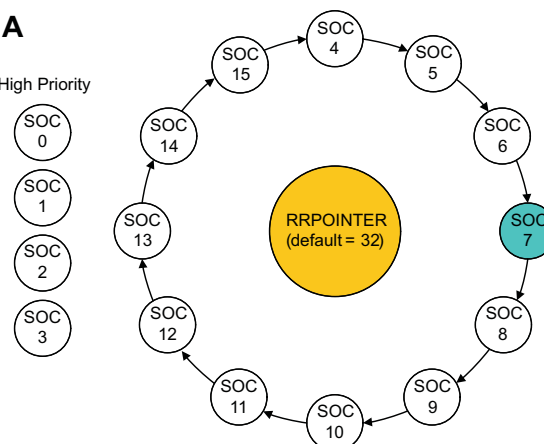
Figure 9-6. High Priority Example

Example when SOC PRIORITY = 4

- A** After reset, SOC4 is 1st on round robin wheel ;
SOC7 receives trigger ;
SOC7 configured channel is converted immediately .
- B** RRPOINTER changes to point to SOC 7 ;
SOC8 is now 1st 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 1st on round robin wheel .

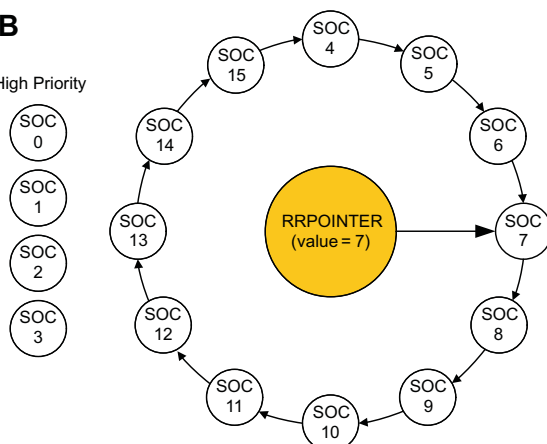
A

High Priority



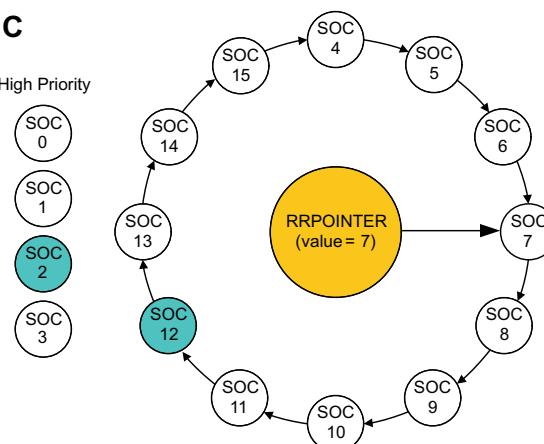
B

High Priority



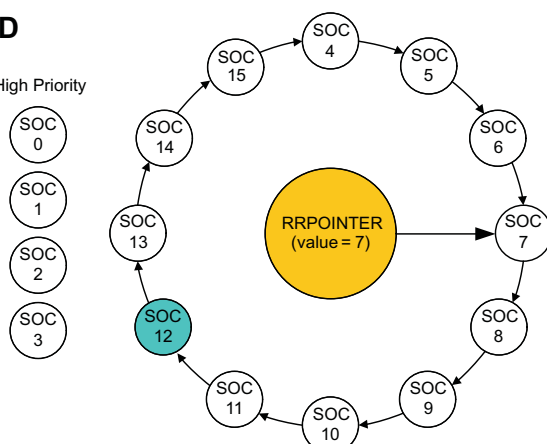
C

High Priority



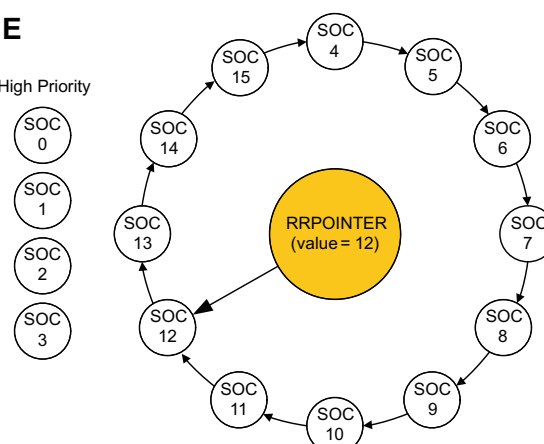
D

High Priority



E

High Priority



9.1.7 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.

9.1.7.1 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 ADCIN3 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.SOCPRCTL.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.

9.1.7.2 Burst Mode Priority Example

An example of priority resolution using burst mode and high-priority SOC's is presented in [Figure 9-7](#).

Figure 9-7. ADC Burst Priority

Example when $SOC_{PRIORITY} = 4$, $BURSTEN = 1$, and $BURSTSIZE = 1$

- A** After reset, SOC4 is 1st 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 1st 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 1st on round robin wheel, waiting for BURSTTRIG.

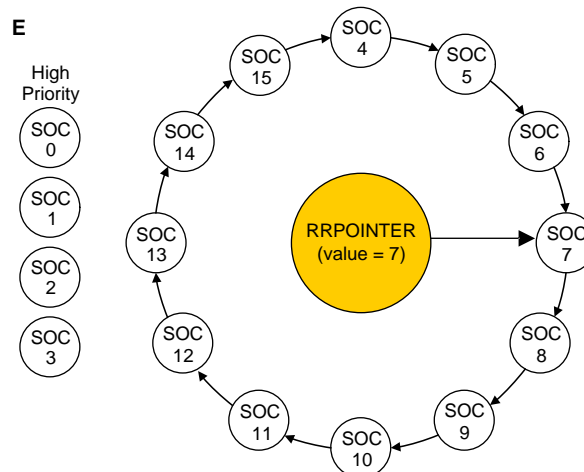
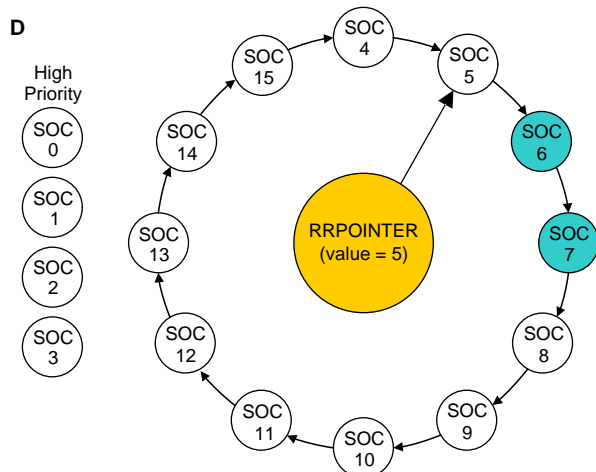
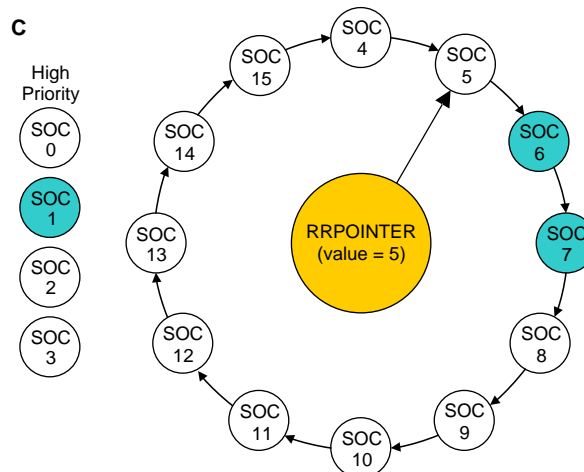
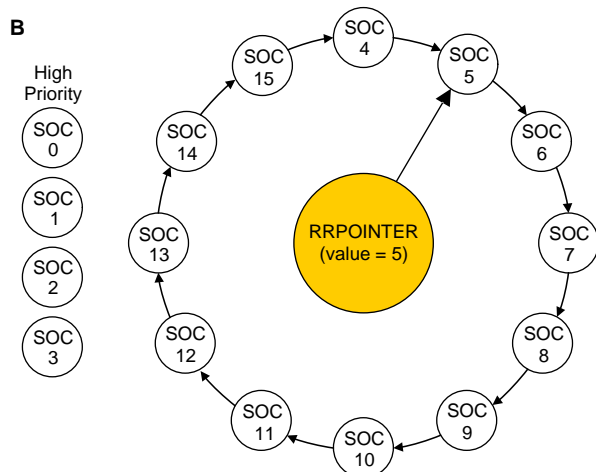
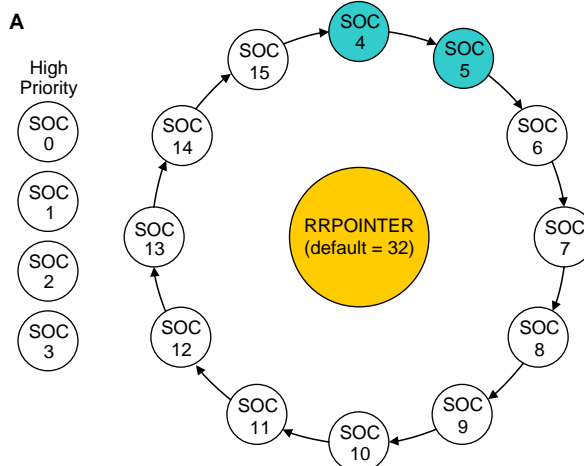
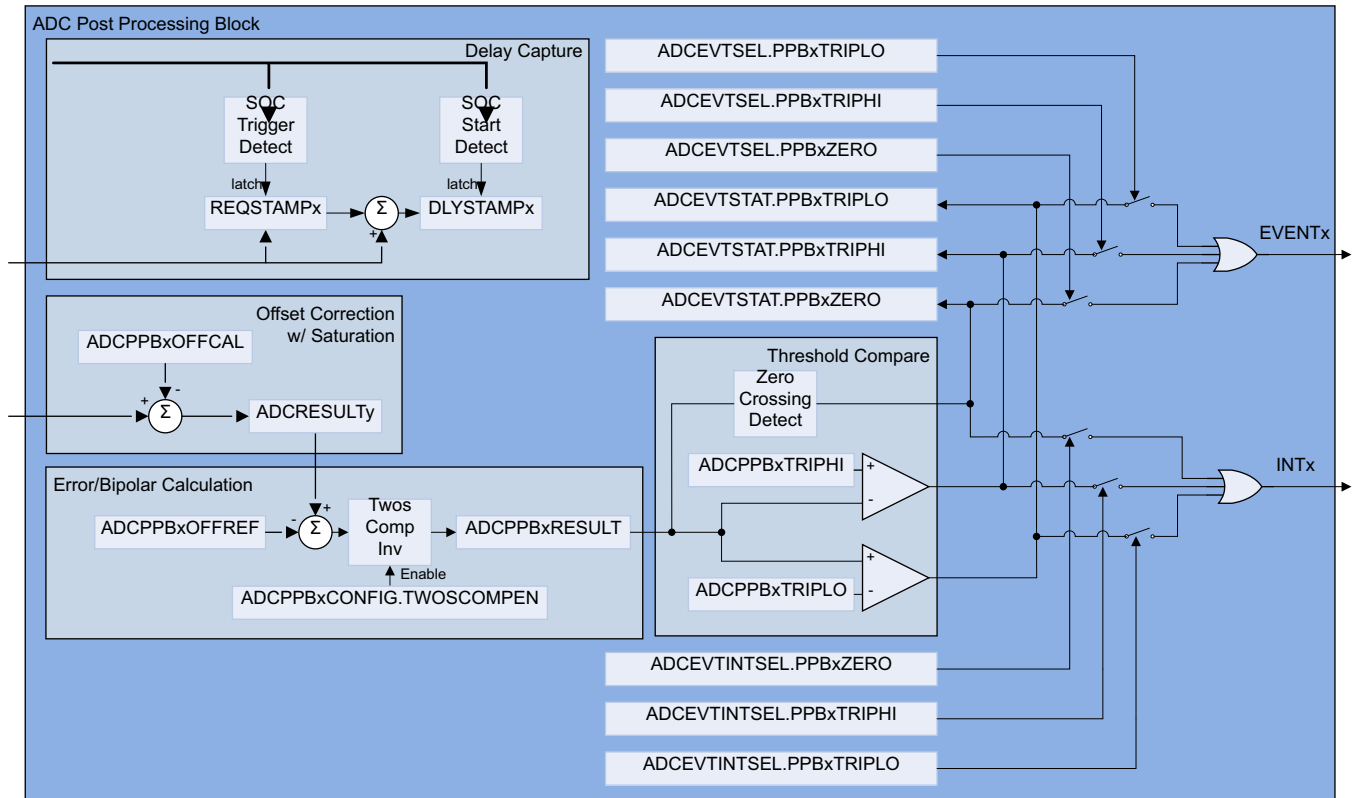


Figure 9-9. ADC PPB Block Diagram


9.1.9.1 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 either 4095 or 65535 on the high end for 12-bit or 16-bit mode, respectively.

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.

9.1.9.2 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:

- In 12-bit mode, 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.

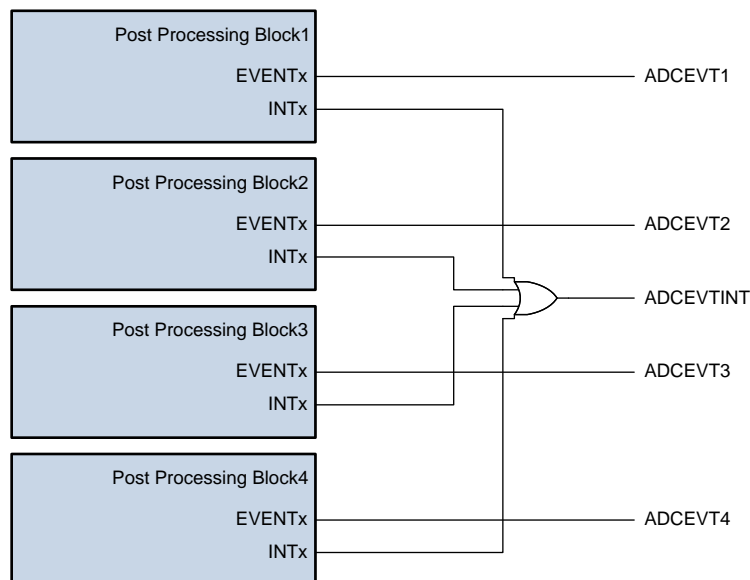
9.1.9.3 PPB Limit Detection and Zero-Crossing Detection

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 (zero-crossing detection does not require further configuration). Whenever these limits are exceeded, the PPBxTRIPHI bit or PPBxTRIPLO bit will be set in the ADCEVTSTAT register. Whenever the ADCPPBxRESULT changes sign, the PPBxZERO bit will be set in the ADCEVTSTAT 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 9-10](#).

Figure 9-10. 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.

9.1.9.4 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

9.1.10 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 at least 1ms delay before sampling.

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.

9.1.11 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. 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 gain and DAC offset from OTP memory to their respective trim registers.
- The CalAdcXINL() functions copy the trim values for linearity from OTP memory to the respective trim registers.
- A different offset trim is required for each possible combination of resolution and signalmode. The GetAdcOffsetTrimOTP(Uint16) function takes an input value corresponding to the ADC, resolution, and signal mode and returns the corresponding offset trim value from OTP memory, which the user then

moves into the ADC offset trim register.

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).

9.1.11.1 ADC Zero Offset Calibration

Zero offset error is defined as the difference from 0 that occurs when converting a voltage at VREFLO (single-ended operation) or the difference from (maximum code/2) when converting ADCINxP = ADCINxN (differential mode). 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 +127 steps (0x7F). 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*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 127 – result from step 3.

Use the following procedure to re-calibrate the ADC offset in 16-bit, differential mode:

1. Set ADCOFFTRIM to no adjustment (0x00).
2. Short ADCINxP and ADCINyN together (external connection) and accumulate some multiple of 16 conversions (e.g. 32*16 conversions = 512 conversions).
3. Divide the accumulated result by the number of conversions (for example, for 512 conversions, divide by 512).
4. Set ADCOFFTRIM to 0 – result from step 3).

9.2 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 in 12-bit mode and 29.5 ADCCLK cycles in 16-bit mode. The exact conversion time is always a whole number of SYSCLK cycles. See the timing diagrams and tables in [Section 9.2.1](#) for exact timings.

9.2.1 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:

- The parameter t_{SH} is the duration of the S+H window. 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 $(ACQPS + 1)$ SYSCLK cycles. ACQPS can be configured individually for each SOC, so t_{SH} will not necessarily be the same for different SOC's.
- The parameter t_{LAT} is the time from the end of the S+H window until the ADC conversion results latch in the ADCRESULTx register. If the ADCRESULTx register is read before this time, the previous conversion results will be returned.
- The parameter t_{EOC} is the time from the end of the S+H window until the next ADC conversion S+H window can begin. In 16-bit mode, this will coincide with the latching of the conversion results, while in 12-bit mode, the subsequent sample can start before the conversion results are latched.
- The parameter t_{INT} is the time from the end of the S+H window until an ADCINT flag is set (if configured). If the INTPULSEPOS bit in the ADCCTL1 register is set, this will coincide with the conversion results being latched into the result register. If the bit is cleared, this will coincide with the end of the S+H window.

Figure 9-11. ADC Timings for 12-bit Mode in Early Interrupt Mode

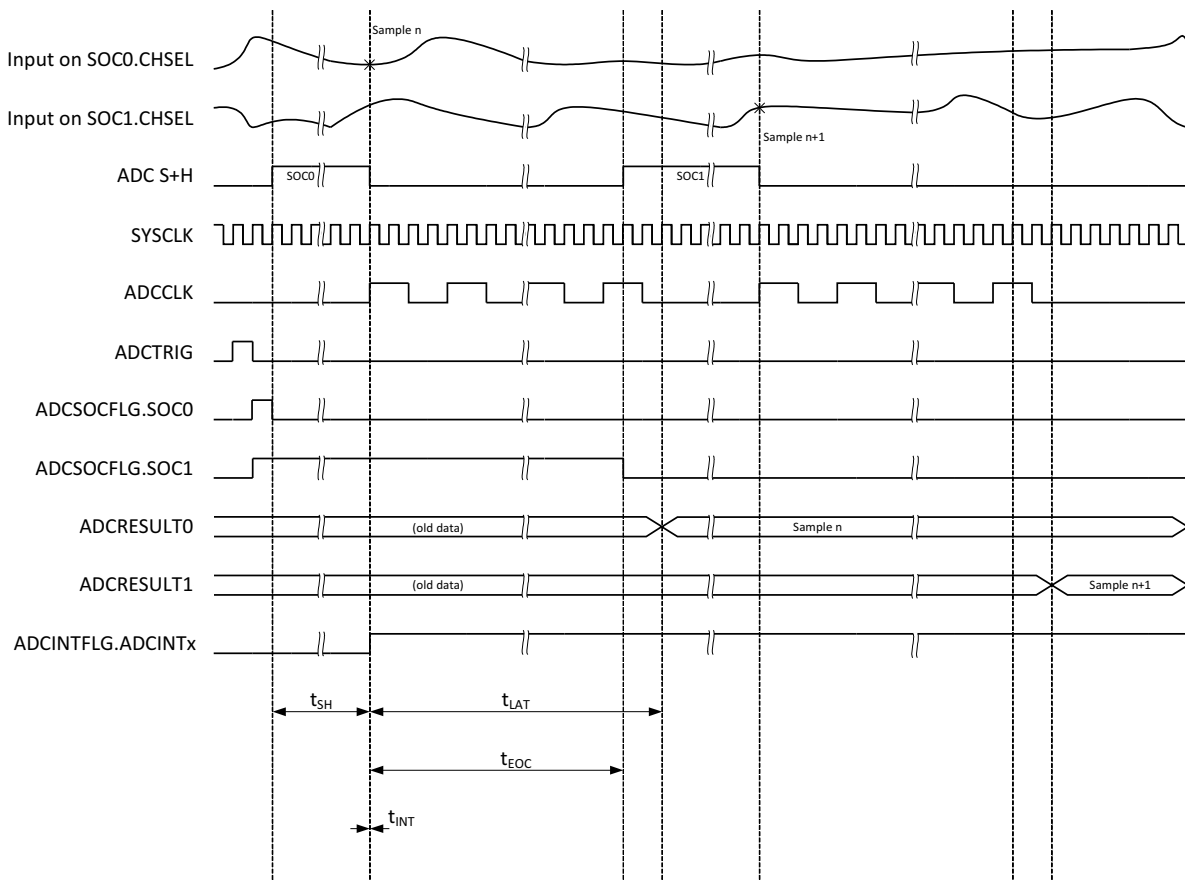


Figure 9-12. ADC Timings for 12-bit Mode in Late Interrupt Mode

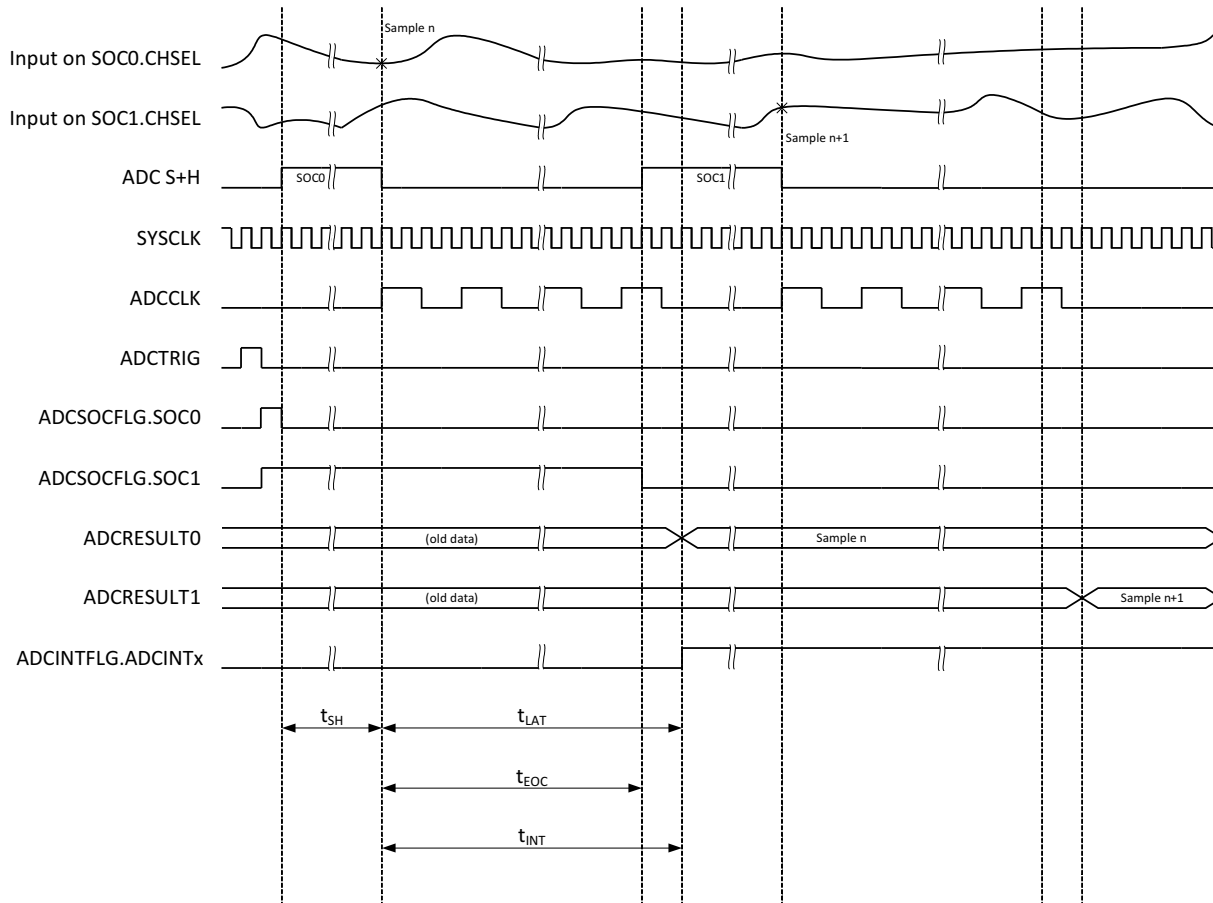


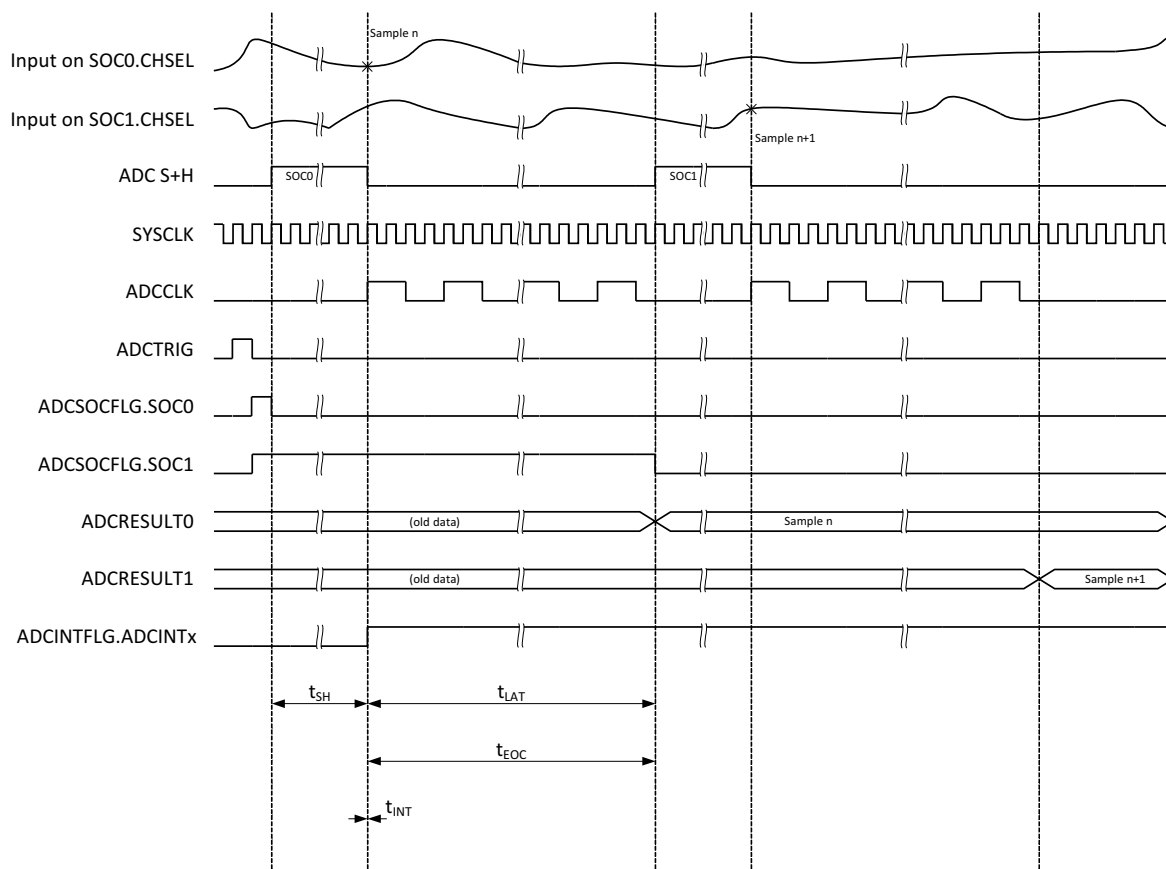
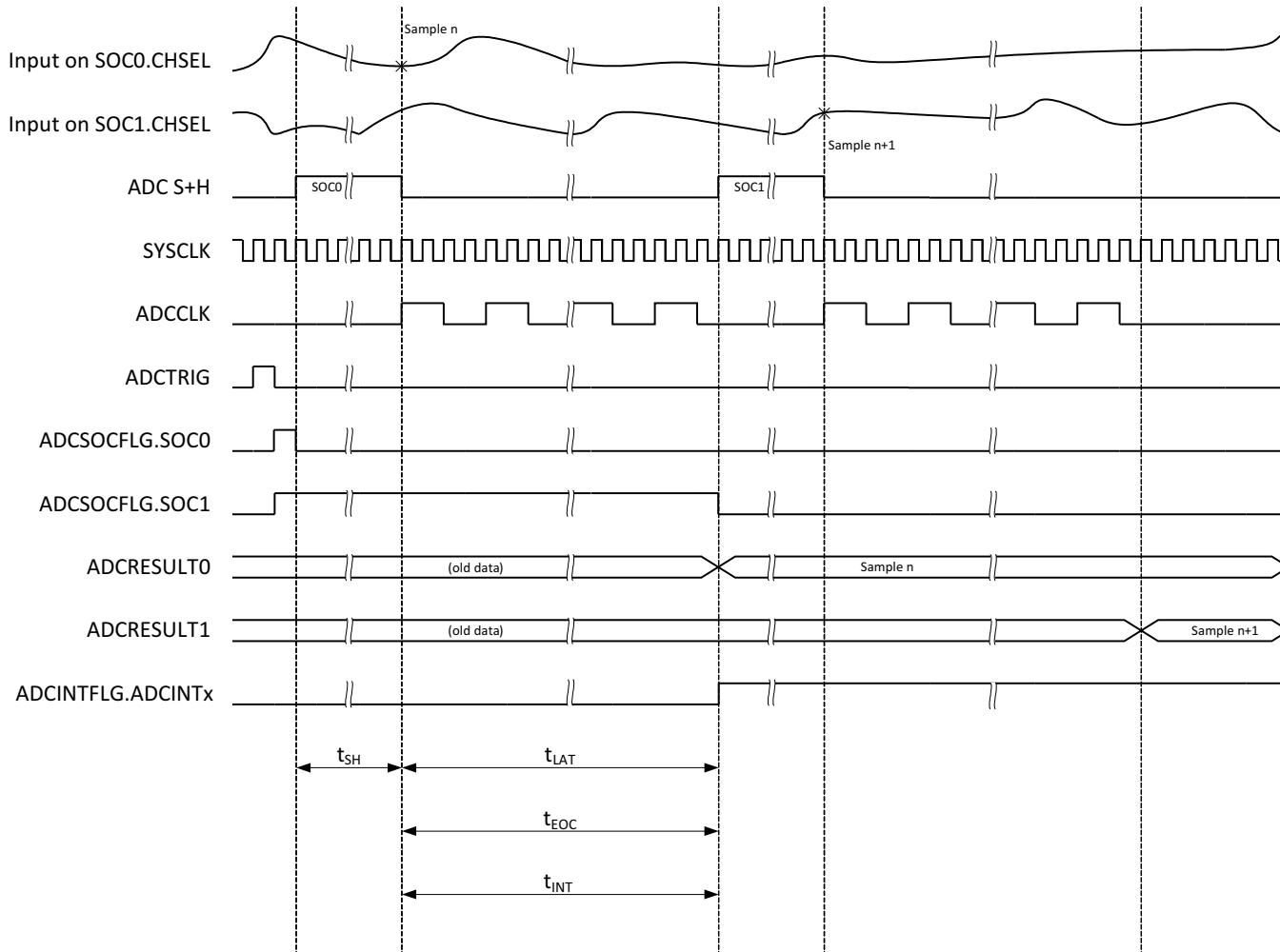
Figure 9-13. ADC Timings for 16-bit Mode in Early Interrupt Mode


Figure 9-14. ADC Timings for 16-bit Mode in Late Interrupt Mode (SYSCLK Cycles)

Table 9-9. 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
3	2.5	26	27	0	27
4	3	31	33	0	33
5	3.5	36	38	0	38
6	4	41	43	0	43
7	4.5	46	48	0	48
8	5	51	54	0	54
9	5.5	56	59	0	59
10	6	61	64	0	64
11	6.5	66	69	0	69
12	7	71	75	0	75
13	7.5	76	80	0	80
14	8	81	85	0	85
15	8.5	86	90	0	90

Table 9-10. ADC Timings in 16-bit Mode

ADCCTL2.PRESCALE	Prescale Ratio	t _{EOC}	t _{LAT}	t _{INT} (Early)	t _{INT} (Late)
0	1	31	31	0	31
2	2	60	60	0	60
3	2.5	75	75	0	75
4	3	90	90	0	90
5	3.5	104	104	0	104
6	4	119	119	0	119
7	4.5	134	134	0	134
8	5	149	149	0	149
9	5.5	163	163	0	163
10	6	178	178	0	178
11	6.5	193	193	0	193
12	7	208	208	0	208
13	7.5	222	222	0	222
14	8	237	237	0	237
15	8.5	252	252	0	252

9.3 Additional Information

The following sections contain additional practical information.

9.3.1 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, Ch. Typically, the S+H duration is chosen such that the sampling capacitor will be charged to within ½ LSB or ¼ LSB of the final value, depending on tolerable settling error. A rough approximation of the required sampling time can be determined using a first-order RC model with R = Rs + Ron and C = Ch. The RC time constant is then (Rs + Ron)·(Ch). The required number of time constants is:

$$t = -\ln\left(\frac{\text{settling accuracy (LSBs)}}{2^N}\right)$$

For example, assuming desired accuracy is ¼ LSB, resolution is 12-bits, Rs = 250Ω, Ron = 300Ω, and Ch = 14.5pF.

$$t = -\ln\left(\frac{\text{settling accuracy (LSBs)}}{2^N}\right) \tau$$

$$t = -\ln\left(\frac{0.25}{2^{12}}\right) (550\text{k}\Omega \cdot 14.5 \text{ pf}) = 77 \text{ ns}$$

The sample and hold window duration for this ADC is

$$T_{S+H} = (ACQPS + 1) \frac{1}{\text{SYSCLK}}$$

If SYSCLK = 200 MHz, then 1/SYSCLK = 5ns, and 1+ACQPS should therefore be chosen to be 16, giving ACQPS = 15.

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.

9.3.2 Achieving Simultaneous Sampling

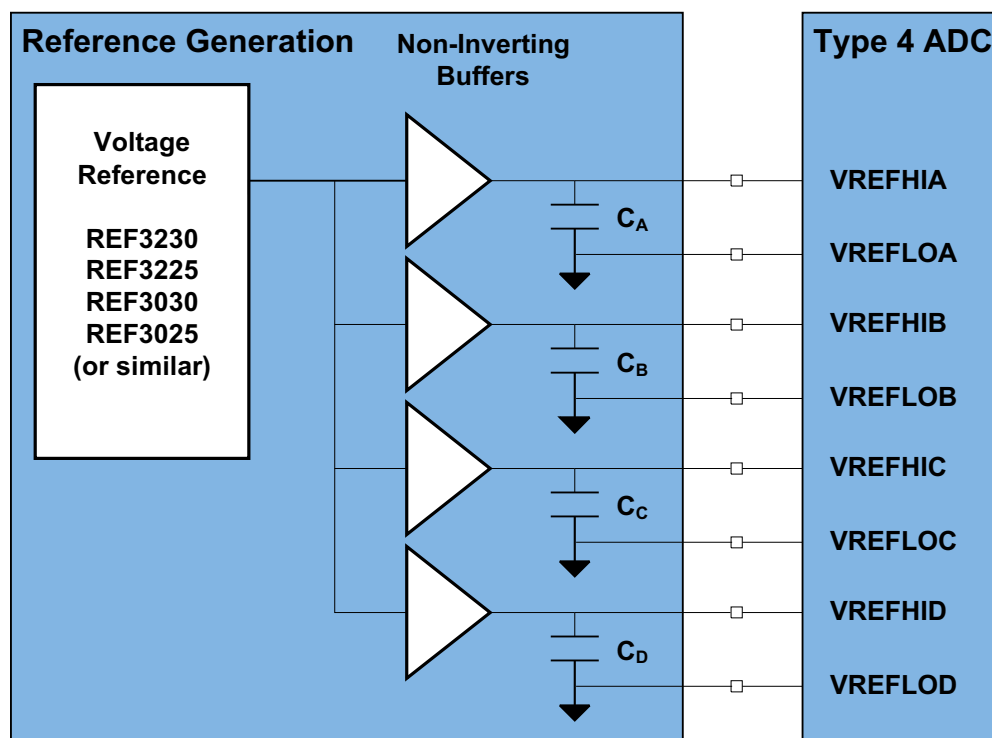
While the Type 4 ADC does not have dual S+H circuits, it is easy to achieve simultaneous sampling. This is accomplished by setting the SOC triggers on two or more ADC modules to use the same trigger source. The example below demonstrates x4 simultaneous sampling based on an ePWM3 event. ADCINA3, ADCINB5, ADCINC5, and ADCIND2 are sampled. An acquisition window of 20 SYSCLK cycles is used, but different durations are possible. SOC0 is used for all ADCs, but a different SOC is possible, including different SOC0s for each ADC.

```
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/D
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/D
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/D
AdcdRegs.ADCSOC0CTL.bit.CHSEL = 2;           //SOC0 will convert ADCIND2
AdcdRegs.ADCSOC0CTL.bit.ACQPS = 19;          //SOC0 will use sample duration of 20 SYSCLK cycles
AdcdRegs.ADCSOC0CTL.bit.TRIGSEL = 10;        //SOC0 will begin conversion on ePWM3 SOCB/D
\
```

When the ePWM3 trigger is received, all four 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.

9.3.3 Designing an External Reference Circuit

One external reference circuit can be shared by one or more ADC module. This is desirable to keep ADC reference mismatch minimal, and to save on the number of components necessary in the system. A block diagram for a shared reference system is presented below in [Figure 9-15](#).

Figure 9-15. Shared Reference System


9.3.4 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 ADCIN13 on ADCA by setting the ENABLE bit in the TSNSCTL register.

NOTE: To sample the temperature sensor, the ADC must be in single-ended 12-bit mode.

9.4 Registers

9.4.1 ADC Base Addresses

Table 9-11. 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
AdcdResultRegs	ADC_RESULT_REGS	0x0000_0B60	0x0000_0B7F
AdcaRegs	ADC_REGS	0x0000_7400	0x0000_747F
AdcbRegs	ADC_REGS	0x0000_7480	0x0000_74FF
AdccRegs	ADC_REGS	0x0000_7500	0x0000_757F
AdcdRegs	ADC_REGS	0x0000_7580	0x0000_75FF

9.4.2 ADC_REGS Registers

Table 9-12 lists the memory-mapped registers for the ADC_REGS. All register offset addresses not listed in Table 9-12 should be considered as reserved locations and the register contents should not be modified.

Table 9-12. ADC_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCCTL1	ADC Control 1 Register	EALLOW	Go
1h	ADCCTL2	ADC Control 2 Register	EALLOW	Go
2h	ADCBURSTCTL	ADC Burst Control Register	EALLOW	Go
3h	ADCINTFLG	ADC Interrupt Flag Register		Go
4h	ADCINTFLGCLR	ADC Interrupt Flag Clear Register		Go
5h	ADCINTOVF	ADC Interrupt Overflow Register		Go
6h	ADCINTOVFCLR	ADC Interrupt Overflow Clear Register		Go
7h	ADCINTSEL1N2	ADC Interrupt 1 and 2 Selection Register	EALLOW	Go
8h	ADCINTSEL3N4	ADC Interrupt 3 and 4 Selection Register	EALLOW	Go
9h	ADCSOCPRCTL	ADC SOC Priority Control Register	EALLOW	Go
Ah	ADCINTSOCSEL1	ADC Interrupt SOC Selection 1 Register	EALLOW	Go
Bh	ADCINTSOCSEL2	ADC Interrupt SOC Selection 2 Register	EALLOW	Go
Ch	ADCSOCFLG1	ADC SOC Flag 1 Register		Go
Dh	ADCSOCFRC1	ADC SOC Force 1 Register		Go
Eh	ADCSOCOVF1	ADC SOC Overflow 1 Register		Go
Fh	ADCSOCOVFCLR1	ADC SOC Overflow Clear 1 Register		Go
10h	ADCSOC0CTL	ADC SOC0 Control Register	EALLOW	Go
12h	ADCSOC1CTL	ADC SOC1 Control Register	EALLOW	Go
14h	ADCSOC2CTL	ADC SOC2 Control Register	EALLOW	Go
16h	ADCSOC3CTL	ADC SOC3 Control Register	EALLOW	Go
18h	ADCSOC4CTL	ADC SOC4 Control Register	EALLOW	Go
1Ah	ADCSOC5CTL	ADC SOC5 Control Register	EALLOW	Go
1Ch	ADCSOC6CTL	ADC SOC6 Control Register	EALLOW	Go
1Eh	ADCSOC7CTL	ADC SOC7 Control Register	EALLOW	Go
20h	ADCSOC8CTL	ADC SOC8 Control Register	EALLOW	Go
22h	ADCSOC9CTL	ADC SOC9 Control Register	EALLOW	Go
24h	ADCSOC10CTL	ADC SOC10 Control Register	EALLOW	Go
26h	ADCSOC11CTL	ADC SOC11 Control Register	EALLOW	Go
28h	ADCSOC12CTL	ADC SOC12 Control Register	EALLOW	Go
2Ah	ADCSOC13CTL	ADC SOC13 Control Register	EALLOW	Go
2Ch	ADCSOC14CTL	ADC SOC14 Control Register	EALLOW	Go
2Eh	ADCSOC15CTL	ADC SOC15 Control Register	EALLOW	Go
30h	ADCEVTSTAT	ADC Event Status Register		Go
32h	ADCEVTCLR	ADC Event Clear Register		Go
34h	ADCEVTSEL	ADC Event Selection Register	EALLOW	Go
36h	ADCEVTINTSEL	ADC Event Interrupt Selection Register	EALLOW	Go
39h	ADCCOUNTER	ADC Counter Register		Go
3Ah	ADCREV	ADC Revision Register		Go
3Bh	ADCOFFTRIM	ADC Offset Trim Register	EALLOW	Go
40h	ADCPPB1CONFIG	ADC PPB1 Config Register	EALLOW	Go
41h	ADCPPB1STAMP	ADC PPB1 Sample Delay Time Stamp Register		Go
42h	ADCPPB1OFFCAL	ADC PPB1 Offset Calibration Register	EALLOW	Go
43h	ADCPPB1OFFREF	ADC PPB1 Offset Reference Register		Go

Table 9-12. ADC_REGS Registers (continued)

Offset	Acronym	Register Name	Write Protection	Section
44h	ADCPPB1TRIPHI	ADC PPB1 Trip High Register	EALLOW	Go
46h	ADCPPB1TRIPLO	ADC PPB1 Trip Low/Trigger Time Stamp Register	EALLOW	Go
48h	ADCPPB2CONFIG	ADC PPB2 Config Register	EALLOW	Go
49h	ADCPPB2STAMP	ADC PPB2 Sample Delay Time Stamp Register		Go
4Ah	ADCPPB2OFFCAL	ADC PPB2 Offset Calibration Register	EALLOW	Go
4Bh	ADCPPB2OFFREF	ADC PPB2 Offset Reference Register		Go
4Ch	ADCPPB2TRIPHI	ADC PPB2 Trip High Register	EALLOW	Go
4Eh	ADCPPB2TRIPLO	ADC PPB2 Trip Low/Trigger Time Stamp Register	EALLOW	Go
50h	ADCPPB3CONFIG	ADC PPB3 Config Register	EALLOW	Go
51h	ADCPPB3STAMP	ADC PPB3 Sample Delay Time Stamp Register		Go
52h	ADCPPB3OFFCAL	ADC PPB3 Offset Calibration Register	EALLOW	Go
53h	ADCPPB3OFFREF	ADC PPB3 Offset Reference Register		Go
54h	ADCPPB3TRIPHI	ADC PPB3 Trip High Register	EALLOW	Go
56h	ADCPPB3TRIPLO	ADC PPB3 Trip Low/Trigger Time Stamp Register	EALLOW	Go
58h	ADCPPB4CONFIG	ADC PPB4 Config Register	EALLOW	Go
59h	ADCPPB4STAMP	ADC PPB4 Sample Delay Time Stamp Register		Go
5Ah	ADCPPB4OFFCAL	ADC PPB4 Offset Calibration Register	EALLOW	Go
5Bh	ADCPPB4OFFREF	ADC PPB4 Offset Reference Register		Go
5Ch	ADCPPB4TRIPHI	ADC PPB4 Trip High Register	EALLOW	Go
5Eh	ADCPPB4TRIPLO	ADC PPB4 Trip Low/Trigger Time Stamp Register	EALLOW	Go
70h	ADCINLTRIM1	ADC Linearity Trim 1 Register	EALLOW	Go
72h	ADCINLTRIM2	ADC Linearity Trim 2 Register	EALLOW	Go
74h	ADCINLTRIM3	ADC Linearity Trim 3 Register	EALLOW	Go
76h	ADCINLTRIM4	ADC Linearity Trim 4 Register	EALLOW	Go
78h	ADCINLTRIM5	ADC Linearity Trim 5 Register	EALLOW	Go
7Ah	ADCINLTRIM6	ADC Linearity Trim 6 Register	EALLOW	Go

9.4.2.1 ADCCTL1 Register (Offset = 0h) [reset = 0h]

ADCCTL1 is shown in [Figure 9-16](#) and described in [Table 9-13](#).

ADC Control 1 Register

Figure 9-16. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-13. 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. 0 ADC is not busy 1 ADC is busy and cannot sample another channel
12	RESERVED	R	0h	Reserved
11-8	ADCBSYCHN	R	0h	ADC Busy Channel. Set when ADC SOC for current channel is generated. When ADCBSY=0: holds the value of the last converted channel When ADCBSY=1: reflects channel currently being processed 0h ADCIN0 is currently processing or was last channel converted 1h ADCIN1 is currently processing or was last channel converted 2h ADCIN2 is currently processing or was last channel converted 3h ADCIN3 is currently processing or was last channel converted 4h ADCIN4 is currently processing or was last channel converted 5h ADCIN5 is currently processing or was last channel converted 6h ADCIN6 is currently processing or was last channel converted 7h ADCIN7 is currently processing or was last channel converted 8h ADCIN8 is currently processing or was last channel converted 9h ADCIN9 is currently processing or was last channel converted Ah ADCIN10 is currently processing or was last channel converted Bh ADCIN11 is currently processing or was last channel converted
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
6-3	RESERVED	R	0h	Reserved
2	INTPULSEPOS	R/W	0h	ADC Interrupt Pulse Position. 0 Interrupt pulse generation occurs at the end of the acquisition window 1 Interrupt pulse generation occurs at the end of the conversion, 1 cycle prior to the ADC result latching into its result register
1-0	RESERVED	R	0h	Reserved

9.4.2.2 ADCCTL2 Register (Offset = 1h) [reset = 0h]

ADCCTL2 is shown in [Figure 9-17](#) and described in [Table 9-14](#).

ADC Control 2 Register

Figure 9-17. ADCCTL2 Register

15	14	13	12	11	10	9	8
RESERVED				RESERVED			
R-0h				R-0h			
7	6	5	4	3	2	1	0
SIGNALMODE	RESOLUTION	RESERVED		PRESCALE			
R/W-0h	R/W-0h	R-0h		R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-14. ADCCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	RESERVED	R	0h	Reserved
7	SIGNALMODE	R/W	0h	SOC Signaling Mode. Selects the input mode of the converter. Use the AdcSetMode function to change the signal mode. 0 Single-ended 1 Differential
6	RESOLUTION	R/W	0h	SOC Conversion Resolution. Selects the resolution of the converter. Use the AdcSetMode function to change the resolution. 0 12-bit resolution 1 16-bit resolution
5-4	RESERVED	R	0h	Reserved
3-0	PRESCALE	R/W	0h	ADC Clock Prescaler. 0000 ADCCLK = Input Clock / 1.0 0001 Invalid 0010 ADCCLK = Input Clock / 2.0 0011 ADCCLK = Input Clock / 2.5 0100 ADCCLK = Input Clock / 3.0 0101 ADCCLK = Input Clock / 3.5 0110 ADCCLK = Input Clock / 4.0 0111 ADCCLK = Input Clock / 4.5 1000 ADCCLK = Input Clock / 5.0 1001 ADCCLK = Input Clock / 5.5 1010 ADCCLK = Input Clock / 6.0 1011 ADCCLK = Input Clock / 6.5 1100 ADCCLK = Input Clock / 7.0 1101 ADCCLK = Input Clock / 7.5 1110 ADCCLK = Input Clock / 8.0 1111 ADCCLK = Input Clock / 8.5

9.4.2.3 ADCBURSTCTL Register (Offset = 2h) [reset = 0h]

ADCBURSTCTL is shown in [Figure 9-18](#) and described in [Table 9-15](#).

ADC Burst Control Register

Figure 9-18. 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					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-15. 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.
14-12	RESERVED	R	0h	Reserved
11-8	BURSTSIZE	R/W	0h	SOC Burst Size Select. This bit field determines how many SOC's 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's converted 2h 3 SOC's converted 3h 4 SOC's converted 4h 5 SOC's converted 5h 6 SOC's converted 6h 7 SOC's converted 7h 8 SOC's converted 8h 9 SOC's converted 9h 10 SOC's converted Ah 11 SOC's converted Bh 12 SOC's converted Ch 13 SOC's converted Dh 14 SOC's converted Eh 15 SOC's converted Fh 16 SOC's converted
7-6	RESERVED	R	0h	Reserved

Table 9-15. 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/C</p> <p>06h BURSTTRIG6 - ePWM1, ADCSOCB/D</p> <p>07h BURSTTRIG7 - ePWM2, ADCSOCA/C</p> <p>08h BURSTTRIG8 - ePWM2, ADCSOCB/D</p> <p>09h BURSTTRIG9 - ePWM3, ADCSOCA/C</p> <p>0Ah BURSTTRIG10 - ePWM3, ADCSOCB/D</p> <p>0Bh BURSTTRIG11 - ePWM4, ADCSOCA/C</p> <p>0Ch BURSTTRIG12 - ePWM4, ADCSOCB/D</p> <p>0Dh BURSTTRIG13 - ePWM5, ADCSOCA/C</p> <p>0Eh BURSTTRIG14 - ePWM5, ADCSOCB/D</p> <p>0Fh BURSTTRIG15 - ePWM6, ADCSOCA/C</p> <p>10h BURSTTRIG16 - ePWM6, ADCSOCB/D</p> <p>11h BURSTTRIG17 - ePWM7, ADCSOCA/C</p> <p>12h BURSTTRIG18 - ePWM7, ADCSOCB/D</p> <p>13h BURSTTRIG19 - ePWM8, ADCSOCA/C</p> <p>14h BURSTTRIG20 - ePWM8, ADCSOCB/D</p> <p>15h BURSTTRIG21 - ePWM9, ADCSOCA/C</p> <p>16h BURSTTRIG22 - ePWM9, ADCSOCB/D</p> <p>17h BURSTTRIG23 - ePWM10, ADCSOCA/C</p> <p>18h BURSTTRIG24 - ePWM10, ADCSOCB/D</p> <p>19h BURSTTRIG25 - ePWM11, ADCSOCA/C</p> <p>1Ah BURSTTRIG26 - ePWM11, ADCSOCB/D</p> <p>1Bh BURSTTRIG27 - ePWM12, ADCSOCA/C</p> <p>1Ch BURSTTRIG28 - ePWM12, ADCSOCB/D</p> <p>1Dh BURSTTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh BURSTTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh BURSTTRIG31 - CPU2 Timer 2, TINT2n</p>

9.4.2.4 ADCINTFLG Register (Offset = 3h) [reset = 0h]

ADCINTFLG is shown in [Figure 9-19](#) and described in [Table 9-16](#).

ADC Interrupt Flag Register

Figure 9-19. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-16. ADCINTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R	0h	ADC Interrupt 4 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear. 0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated If the ADC interrupt is placed in continuous 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.
2	ADCINT3	R	0h	ADC Interrupt 3 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear. 0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated If the ADC interrupt is placed in continuous 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.
1	ADCINT2	R	0h	ADC Interrupt 2 Flag. Reading these flags indicates if the associated ADCINT pulse was generated since the last clear. 0 No ADC interrupt pulse generated 1 ADC interrupt pulse generated If the ADC interrupt is placed in continuous 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.

Table 9-16. 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</p> <p>1 ADC interrupt pulse generated</p> <p>If the ADC interrupt is placed in continuous 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>

9.4.2.5 ADCINTFLGCLR Register (Offset = 4h) [reset = 0h]

ADCINTFLGCLR is shown in [Figure 9-20](#) and described in [Table 9-17](#).

ADC Interrupt Flag Clear Register

Figure 9-20. 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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-17. ADCINTFLGCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R=0/W=1	0h	<p>ADC Interrupt 4 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"> 1. SW has priority and will clear the flag 2. HW set will be discarded <p>no signal will propagate to the PIE from the latch</p> <p>3. Overflow flag/condition will be generated</p>
2	ADCINT3	R=0/W=1	0h	<p>ADC Interrupt 3 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"> 1. SW has priority and will clear the flag 2. HW set will be discarded <p>no signal will propagate to the PIE from the latch</p> <p>3. Overflow flag/condition will be generated</p>
1	ADCINT2	R=0/W=1	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"> 1. SW has priority and will clear the flag 2. HW set will be discarded <p>no signal will propagate to the PIE from the latch</p> <p>3. Overflow flag/condition will be generated</p>

Table 9-17. ADCINTFLGCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	ADCINT1	R=0/W=1	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"> 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

9.4.2.6 ADCINTOVF Register (Offset = 5h) [reset = 0h]

ADCINTOVF is shown in [Figure 9-21](#) and described in [Table 9-18](#).

ADC Interrupt Overflow Register

Figure 9-21. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-18. ADCINTOVF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R	0h	<p>ADC Interrupt 4 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.</p> <p>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>
2	ADCINT3	R	0h	<p>ADC Interrupt 3 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.</p> <p>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>
1	ADCINT2	R	0h	<p>ADC Interrupt 2 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.</p> <p>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>
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.</p> <p>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>

9.4.2.7 ADCINTOVFCLR Register (Offset = 6h) [reset = 0h]

ADCINTOVFCLR is shown in [Figure 9-22](#) and described in [Table 9-19](#).

ADC Interrupt Overflow Clear Register

Figure 9-22. 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/W=1-0h	R=0/W=1-0h	R=0/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-19. ADCINTOVFCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R=0/W=1	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.
2	ADCINT3	R=0/W=1	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.
1	ADCINT2	R=0/W=1	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.
0	ADCINT1	R=0/W=1	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.

9.4.2.8 ADCINTSEL1N2 Register (Offset = 7h) [reset = 0h]

ADCINTSEL1N2 is shown in [Figure 9-23](#) and described in [Table 9-20](#).

ADC Interrupt 1 and 2 Selection Register

Figure 9-23. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-20. ADCINTSEL1N2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT2CONT	R/W	0h	ADCINT2 Continuous Mode Enable 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.
13	INT2E	R/W	0h	ADCINT2 Interrupt Enable 0 ADCINT2 is disabled 1 ADCINT2 is enabled
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
7	RESERVED	R	0h	Reserved
6	INT1CONT	R/W	0h	ADCINT1 Continuous Mode Enable 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.

Table 9-20. ADCINTSEL1N2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	INT1E	R/W	0h	ADCINT1 Interrupt Enable 0 ADCINT1 is disabled 1 ADCINT1 is enabled
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

9.4.2.9 ADCINTSEL3N4 Register (Offset = 8h) [reset = 0h]

ADCINTSEL3N4 is shown in [Figure 9-24](#) and described in [Table 9-21](#).

ADC Interrupt 3 and 4 Selection Register

Figure 9-24. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-21. ADCINTSEL3N4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT4CONT	R/W	0h	ADCINT4 Continuous Mode Enable 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.
13	INT4E	R/W	0h	ADCINT4 Interrupt Enable 0 ADCINT4 is disabled 1 ADCINT4 is enabled
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
7	RESERVED	R	0h	Reserved
6	INT3CONT	R/W	0h	ADCINT3 Continuous Mode Enable 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.

Table 9-21. ADCINTSEL3N4 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	INT3E	R/W	0h	ADCINT3 Interrupt Enable 0 ADCINT3 is disabled 1 ADCINT3 is enabled
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

9.4.2.10 ADCSOCPRCTL Register (Offset = 9h) [reset = 200h]

ADCSOCPRCTL is shown in [Figure 9-25](#) and described in [Table 9-22](#).

ADC SOC Priority Control Register

Figure 9-25. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-22. ADCSOCPRCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved

Table 9-22. ADCSOCPRCTL 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, SOC1 is highest round robin priority.</p> <p>02h SOC2 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>03h SOC3 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>04h SOC4 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>05h SOC5 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>06h SOC6 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>07h SOC7 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>08h SOC8 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>09h SOC9 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Ah SOC10 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Bh SOC11 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Ch SOC12 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Dh SOC13 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Eh SOC14 was last round robin SOC to convert, SOC1 is highest round robin priority.</p> <p>0Fh SOC15 was last round robin SOC to convert, SOC1 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 SOCPRCTL 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>

Table 9-22. 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>

9.4.2.11 ADCINTSOCSEL1 Register (Offset = Ah) [reset = 0h]

ADCINTSOCSEL1 is shown in [Figure 9-26](#) and described in [Table 9-23](#).

ADC Interrupt SOC Selection 1 Register

Figure 9-26. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-23. 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. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC7. TRIGSEL field determines SOC7 trigger. 01 ADCINT1 will trigger SOC7. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC7. TRIGSEL field is ignored. 11 Invalid selection.
13-12	SOC6	R/W	0h	SOC6 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC6. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC6. TRIGSEL field determines SOC6 trigger. 01 ADCINT1 will trigger SOC6. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC6. TRIGSEL field is ignored. 11 Invalid selection.
11-10	SOC5	R/W	0h	SOC5 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC5. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC5. TRIGSEL field determines SOC5 trigger. 01 ADCINT1 will trigger SOC5. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC5. TRIGSEL field is ignored. 11 Invalid selection.
9-8	SOC4	R/W	0h	SOC4 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC4. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC4. TRIGSEL field determines SOC4 trigger. 01 ADCINT1 will trigger SOC4. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC4. TRIGSEL field is ignored. 11 Invalid selection.

Table 9-23. 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. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC3. TRIGSEL field determines SOC3 trigger.</p> <p>01 ADCINT1 will trigger SOC3. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC3. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
5-4	SOC2	R/W	0h	<p>SOC2 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC2. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC2. TRIGSEL field determines SOC2 trigger.</p> <p>01 ADCINT1 will trigger SOC2. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC2. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
3-2	SOC1	R/W	0h	<p>SOC1 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC1. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC1. TRIGSEL field determines SOC1 trigger.</p> <p>01 ADCINT1 will trigger SOC1. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC1. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
1-0	SOC0	R/W	0h	<p>SOC0 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC0. This field is OR'ed with the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC0. TRIGSEL field determines SOC0 trigger.</p> <p>01 ADCINT1 will trigger SOC0. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC0. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>

9.4.2.12 ADCINTSOCSEL2 Register (Offset = Bh) [reset = 0h]

ADCINTSOCSEL2 is shown in [Figure 9-27](#) and described in [Table 9-24](#).

ADC Interrupt SOC Selection 2 Register

Figure 9-27. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-24. 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. This field overrides the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC15. TRIGSEL field determines SOC15 trigger. 01 ADCINT1 will trigger SOC15. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC15. TRIGSEL field is ignored. 11 Invalid selection.
13-12	SOC14	R/W	0h	SOC14 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC14. This field overrides the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC14. TRIGSEL field determines SOC14 trigger. 01 ADCINT1 will trigger SOC14. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC14. TRIGSEL field is ignored. 11 Invalid selection.
11-10	SOC13	R/W	0h	SOC13 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC13. This field overrides the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC13. TRIGSEL field determines SOC13 trigger. 01 ADCINT1 will trigger SOC13. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC13. TRIGSEL field is ignored. 11 Invalid selection.
9-8	SOC12	R/W	0h	SOC12 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC12. This field overrides the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC12. TRIGSEL field determines SOC12 trigger. 01 ADCINT1 will trigger SOC12. TRIGSEL field is ignored. 10 ADCINT2 will trigger SOC12. TRIGSEL field is ignored. 11 Invalid selection.

Table 9-24. 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. This field overrides the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC11. TRIGSEL field determines SOC11 trigger.</p> <p>01 ADCINT1 will trigger SOC11. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC11. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
5-4	SOC10	R/W	0h	<p>SOC10 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC10. This field overrides the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC10. TRIGSEL field determines SOC10 trigger.</p> <p>01 ADCINT1 will trigger SOC10. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC10. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
3-2	SOC9	R/W	0h	<p>SOC9 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC9. This field overrides the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC9. TRIGSEL field determines SOC9 trigger.</p> <p>01 ADCINT1 will trigger SOC9. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC9. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>
1-0	SOC8	R/W	0h	<p>SOC8 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC8. This field overrides the TRIGSEL field in the ADCSOCxCTL register.</p> <p>00 No ADCINT will trigger SOC8. TRIGSEL field determines SOC8 trigger.</p> <p>01 ADCINT1 will trigger SOC8. TRIGSEL field is ignored.</p> <p>10 ADCINT2 will trigger SOC8. TRIGSEL field is ignored.</p> <p>11 Invalid selection.</p>

9.4.2.13 ADCSOCFLG1 Register (Offset = Ch) [reset = 0h]

ADCSOCFLG1 is shown in [Figure 9-28](#) and described in [Table 9-25](#).

ADC SOC Flag 1 Register

Figure 9-28. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-25. ADCSOCFLG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SOC15	R	0h	SOC15 Start of Conversion Flag. Indicates the state of SOC15 conversions. 0 No sample pending for SOC15. 1 Trigger has been received and sample is pending for SOC15. 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.
14	SOC14	R	0h	SOC14 Start of Conversion Flag. Indicates the state of SOC14 conversions. 0 No sample pending for SOC14. 1 Trigger has been received and sample is pending for SOC14. 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.
13	SOC13	R	0h	SOC13 Start of Conversion Flag. Indicates the state of SOC13 conversions. 0 No sample pending for SOC13. 1 Trigger has been received and sample is pending for SOC13. 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.
12	SOC12	R	0h	SOC12 Start of Conversion Flag. Indicates the state of SOC12 conversions. 0 No sample pending for SOC12. 1 Trigger has been received and sample is pending for SOC12. 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.

Table 9-25. ADCSOCFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	SOC11	R	0h	<p>SOC11 Start of Conversion Flag. Indicates the state of SOC11 conversions.</p> <p>0 No sample pending for SOC11.</p> <p>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>
10	SOC10	R	0h	<p>SOC10 Start of Conversion Flag. Indicates the state of SOC10 conversions.</p> <p>0 No sample pending for SOC10.</p> <p>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>
9	SOC9	R	0h	<p>SOC9 Start of Conversion Flag. Indicates the state of SOC9 conversions.</p> <p>0 No sample pending for SOC9.</p> <p>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>
8	SOC8	R	0h	<p>SOC8 Start of Conversion Flag. Indicates the state of SOC8 conversions.</p> <p>0 No sample pending for SOC8.</p> <p>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>
7	SOC7	R	0h	<p>SOC7 Start of Conversion Flag. Indicates the state of SOC7 conversions.</p> <p>0 No sample pending for SOC7.</p> <p>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>

Table 9-25. ADCSOCFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	SOC6	R	0h	<p>SOC6 Start of Conversion Flag. Indicates the state of SOC6 conversions.</p> <p>0 No sample pending for SOC6.</p> <p>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>
5	SOC5	R	0h	<p>SOC5 Start of Conversion Flag. Indicates the state of SOC5 conversions.</p> <p>0 No sample pending for SOC5.</p> <p>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>
4	SOC4	R	0h	<p>SOC4 Start of Conversion Flag. Indicates the state of SOC4 conversions.</p> <p>0 No sample pending for SOC4.</p> <p>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>
3	SOC3	R	0h	<p>SOC3 Start of Conversion Flag. Indicates the state of SOC3 conversions.</p> <p>0 No sample pending for SOC3.</p> <p>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>
2	SOC2	R	0h	<p>SOC2 Start of Conversion Flag. Indicates the state of SOC2 conversions.</p> <p>0 No sample pending for SOC2.</p> <p>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>

Table 9-25. ADCSOCFLG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	SOC1	R	0h	<p>SOC1 Start of Conversion Flag. Indicates the state of SOC1 conversions.</p> <p>0 No sample pending for SOC1.</p> <p>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>
0	SOC0	R	0h	<p>SOC0 Start of Conversion Flag. Indicates the state of SOC0 conversions.</p> <p>0 No sample pending for SOC0.</p> <p>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>

9.4.2.14 ADCSOCFRC1 Register (Offset = Dh) [reset = 0h]

ADCSOCFRC1 is shown in [Figure 9-29](#) and described in [Table 9-26](#).

ADC SOC Force 1 Register

Figure 9-29. ADCSOCFRC1 Register

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
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
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-26. ADCSOCFRC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SOC15	R=0/W=1	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>
14	SOC14	R=0/W=1	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>
13	SOC13	R=0/W=1	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>

Table 9-26. ADCSOCFRC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12	SOC12	R=0/W=1	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>
11	SOC11	R=0/W=1	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>
10	SOC10	R=0/W=1	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>
9	SOC9	R=0/W=1	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>

Table 9-26. ADCSOCFRC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	SOC8	R=0/W=1	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>
7	SOC7	R=0/W=1	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>
6	SOC6	R=0/W=1	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>
5	SOC5	R=0/W=1	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>

Table 9-26. ADCSOCFRC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	SOC4	R=0/W=1	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>
3	SOC3	R=0/W=1	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>
2	SOC2	R=0/W=1	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>
1	SOC1	R=0/W=1	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.</p> <p>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>

Table 9-26. ADCSOCFRC1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	SOC0	R=0/W=1	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.</p> <p>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>

9.4.2.15 ADCSOCOVF1 Register (Offset = Eh) [reset = 0h]

ADCSOCOVF1 is shown in [Figure 9-30](#) and described in [Table 9-27](#).

ADC SOC Overflow 1 Register

Figure 9-30. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-27. ADCSOCOVF1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SOC15	R	0h	SOC15 Start of Conversion Overflow Flag. Indicates an SOC15 event was generated in hardware while an existing SOC15 event was already pending. 0 No SOC15 event overflow. 1 SOC15 event overflow. 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.
14	SOC14	R	0h	SOC14 Start of Conversion Overflow Flag. Indicates an SOC14 event was generated in hardware while an existing SOC14 event was already pending. 0 No SOC14 event overflow. 1 SOC14 event overflow. 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.
13	SOC13	R	0h	SOC13 Start of Conversion Overflow Flag. Indicates an SOC13 event was generated in hardware while an existing SOC13 event was already pending. 0 No SOC13 event overflow. 1 SOC13 event overflow. 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.
12	SOC12	R	0h	SOC12 Start of Conversion Overflow Flag. Indicates an SOC12 event was generated in hardware while an existing SOC12 event was already pending. 0 No SOC12 event overflow. 1 SOC12 event overflow. 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.

Table 9-27. ADCSOCOVF1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>

Table 9-27. ADCSOCOVF1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>
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.</p> <p>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>

9.4.2.16 ADCSOCOVFCLR1 Register (Offset = Fh) [reset = 0h]

ADCSOCOVFCLR1 is shown in [Figure 9-31](#) and described in [Table 9-28](#).

ADC SOC Overflow Clear 1 Register

Figure 9-31. ADCSOCOVFCLR1 Register

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
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
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-28. ADCSOCOVFCLR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SOC15	R=0/W=1	0h	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. 0 No action. 1 Clear SOC15 overflow flag. 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..
14	SOC14	R=0/W=1	0h	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. 0 No action. 1 Clear SOC14 overflow flag. 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..
13	SOC13	R=0/W=1	0h	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. 0 No action. 1 Clear SOC13 overflow flag. 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..
12	SOC12	R=0/W=1	0h	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. 0 No action. 1 Clear SOC12 overflow flag. 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..
11	SOC11	R=0/W=1	0h	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. 0 No action. 1 Clear SOC11 overflow flag. 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..

Table 9-28. ADCSOCOVCLR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	SOC10	R=0/W=1	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.</p> <p>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>
9	SOC9	R=0/W=1	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.</p> <p>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>
8	SOC8	R=0/W=1	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.</p> <p>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>
7	SOC7	R=0/W=1	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.</p> <p>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>
6	SOC6	R=0/W=1	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.</p> <p>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>
5	SOC5	R=0/W=1	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.</p> <p>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>
4	SOC4	R=0/W=1	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.</p> <p>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>

Table 9-28. ADCSOCOVFCLR1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	SOC3	R=0/W=1	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.</p> <p>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>
2	SOC2	R=0/W=1	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.</p> <p>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>
1	SOC1	R=0/W=1	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.</p> <p>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>
0	SOC0	R=0/W=1	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.</p> <p>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>

9.4.2.17 ADCSOC0CTL Register (Offset = 10h) [reset = 0h]

ADCSOC0CTL is shown in [Figure 9-32](#) and described in [Table 9-29](#).

ADC SOC0 Control Register

Figure 9-32. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-29. ADCSOC0CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-29. ADCSOC0CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-29. ADCSOC0CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.18 ADCSOC1CTL Register (Offset = 12h) [reset = 0h]

ADCSOC1CTL is shown in [Figure 9-33](#) and described in [Table 9-30](#).

ADC SOC1 Control Register

Figure 9-33. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-30. ADCSOC1CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-30. ADCSOC1CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-30. ADCSOC1CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.19 ADCSOC2CTL Register (Offset = 14h) [reset = 0h]

ADCSOC2CTL is shown in [Figure 9-34](#) and described in [Table 9-31](#).

ADC SOC2 Control Register

Figure 9-34. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-31. ADCSOC2CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-31. ADCSOC2CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-31. ADCSOC2CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.20 ADCSOC3CTL Register (Offset = 16h) [reset = 0h]

ADCSOC3CTL is shown in [Figure 9-35](#) and described in [Table 9-32](#).

ADC SOC3 Control Register

Figure 9-35. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-32. ADCSOC3CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-32. ADCSOC3CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-32. ADCSOC3CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.21 ADCSOC4CTL Register (Offset = 18h) [reset = 0h]

ADCSOC4CTL is shown in [Figure 9-36](#) and described in [Table 9-33](#).

ADC SOC4 Control Register

Figure 9-36. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-33. ADCSOC4CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-33. ADCSOC4CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>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 ADCTRIG21 - ePWM9, ADCSOCA 16h ADCTRIG22 - ePWM9, ADCSOCA 17h ADCTRIG23 - ePWM10, ADCSOCA 18h ADCTRIG24 - ePWM10, ADCSOCA 19h ADCTRIG25 - ePWM11, ADCSOCA 1Ah ADCTRIG26 - ePWM11, ADCSOCA 1Bh ADCTRIG27 - ePWM12, ADCSOCA 1Ch ADCTRIG28 - ePWM12, ADCSOCA 1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n 1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n 1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-33. ADCSOC4CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.22 ADCSOC5CTL Register (Offset = 1Ah) [reset = 0h]

ADCSOC5CTL is shown in [Figure 9-37](#) and described in [Table 9-34](#).

ADC SOC5 Control Register

Figure 9-37. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-34. ADCSOC5CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-34. ADCSOC5CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-34. ADCSOC5CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.23 ADCSOC6CTL Register (Offset = 1Ch) [reset = 0h]

ADCSOC6CTL is shown in [Figure 9-38](#) and described in [Table 9-35](#).

ADC SOC6 Control Register

Figure 9-38. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-35. ADCSOC6CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-35. ADCSOC6CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>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 ADCTRIG21 - ePWM9, ADCSOCA 16h ADCTRIG22 - ePWM9, ADCSOCA 17h ADCTRIG23 - ePWM10, ADCSOCA 18h ADCTRIG24 - ePWM10, ADCSOCA 19h ADCTRIG25 - ePWM11, ADCSOCA 1Ah ADCTRIG26 - ePWM11, ADCSOCA 1Bh ADCTRIG27 - ePWM12, ADCSOCA 1Ch ADCTRIG28 - ePWM12, ADCSOCA 1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n 1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n 1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-35. ADCSOC6CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.24 ADCSOC7CTL Register (Offset = 1Eh) [reset = 0h]

ADCSOC7CTL is shown in [Figure 9-39](#) and described in [Table 9-36](#).

ADC SOC7 Control Register

Figure 9-39. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-36. ADCSOC7CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-36. ADCSOC7CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>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 ADCTRIG21 - ePWM9, ADCSOCA 16h ADCTRIG22 - ePWM9, ADCSOCA 17h ADCTRIG23 - ePWM10, ADCSOCA 18h ADCTRIG24 - ePWM10, ADCSOCA 19h ADCTRIG25 - ePWM11, ADCSOCA 1Ah ADCTRIG26 - ePWM11, ADCSOCA 1Bh ADCTRIG27 - ePWM12, ADCSOCA 1Ch ADCTRIG28 - ePWM12, ADCSOCA 1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n 1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n 1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-36. ADCSOC7CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.25 ADCSOC8CTL Register (Offset = 20h) [reset = 0h]

ADCSOC8CTL is shown in [Figure 9-40](#) and described in [Table 9-37](#).

ADC SOC8 Control Register

Figure 9-40. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-37. ADCSOC8CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-37. ADCSOC8CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-37. ADCSOC8CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.26 ADCSOC9CTL Register (Offset = 22h) [reset = 0h]

ADCSOC9CTL is shown in [Figure 9-41](#) and described in [Table 9-38](#).

ADC SOC9 Control Register

Figure 9-41. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-38. ADCSOC9CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-38. ADCSOC9CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-38. ADCSOC9CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.27 ADCSOC10CTL Register (Offset = 24h) [reset = 0h]

ADCSOC10CTL is shown in [Figure 9-42](#) and described in [Table 9-39](#).

ADC SOC10 Control Register

Figure 9-42. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-39. ADCSOC10CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-39. ADCSOC10CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-39. ADCSOC10CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.28 ADCSOC11CTL Register (Offset = 26h) [reset = 0h]

ADCSOC11CTL is shown in [Figure 9-43](#) and described in [Table 9-40](#).

ADC SOC11 Control Register

Figure 9-43. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-40. ADCSOC11CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-40. ADCSOC11CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>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 ADCTRIG21 - ePWM9, ADCSOCA 16h ADCTRIG22 - ePWM9, ADCSOCA 17h ADCTRIG23 - ePWM10, ADCSOCA 18h ADCTRIG24 - ePWM10, ADCSOCA 19h ADCTRIG25 - ePWM11, ADCSOCA 1Ah ADCTRIG26 - ePWM11, ADCSOCA 1Bh ADCTRIG27 - ePWM12, ADCSOCA 1Ch ADCTRIG28 - ePWM12, ADCSOCA 1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n 1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n 1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-40. ADCSOC11CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.29 ADCSOC12CTL Register (Offset = 28h) [reset = 0h]

ADCSOC12CTL is shown in [Figure 9-44](#) and described in [Table 9-41](#).

ADC SOC12 Control Register

Figure 9-44. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-41. ADCSOC12CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-41. ADCSOC12CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>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 ADCTRIG21 - ePWM9, ADCSOCA 16h ADCTRIG22 - ePWM9, ADCSOCA 17h ADCTRIG23 - ePWM10, ADCSOCA 18h ADCTRIG24 - ePWM10, ADCSOCA 19h ADCTRIG25 - ePWM11, ADCSOCA 1Ah ADCTRIG26 - ePWM11, ADCSOCA 1Bh ADCTRIG27 - ePWM12, ADCSOCA 1Ch ADCTRIG28 - ePWM12, ADCSOCA 1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n 1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n 1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-41. ADCSOC12CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.30 ADCSOC13CTL Register (Offset = 2Ah) [reset = 0h]

ADCSOC13CTL is shown in [Figure 9-45](#) and described in [Table 9-42](#).

ADC SOC13 Control Register

Figure 9-45. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-42. ADCSOC13CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-42. ADCSOC13CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-42. ADCSOC13CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.31 ADCSOC14CTL Register (Offset = 2Ch) [reset = 0h]

ADCSOC14CTL is shown in [Figure 9-46](#) and described in [Table 9-43](#).

ADC SOC14 Control Register

Figure 9-46. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-43. ADCSOC14CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-43. ADCSOC14CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-43. ADCSOC14CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.32 ADCSOC15CTL Register (Offset = 2Eh) [reset = 0h]

ADCSOC15CTL is shown in [Figure 9-47](#) and described in [Table 9-44](#).

ADC SOC15 Control Register

Figure 9-47. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-44. ADCSOC15CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved

Table 9-44. ADCSOC15CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24-20	TRIGSEL	R/W	0h	<p>SOC Trigger Source Select. Along with the corresponding SOC field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the corresponding SOC flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>00h ADCTRIG0 - Software only</p> <p>01h ADCTRIG1 - CPU1 Timer 0, TINT0n</p> <p>02h ADCTRIG2 - CPU1 Timer 1, TINT1n</p> <p>03h ADCTRIG3 - CPU1 Timer 2, TINT2n</p> <p>04h ADCTRIG4 - GPIO, ADCEXTSOC</p> <p>05h ADCTRIG5 - ePWM1, ADCSOCA</p> <p>06h ADCTRIG6 - ePWM1, ADCSOCA</p> <p>07h ADCTRIG7 - ePWM2, ADCSOCA</p> <p>08h ADCTRIG8 - ePWM2, ADCSOCA</p> <p>09h ADCTRIG9 - ePWM3, ADCSOCA</p> <p>0Ah ADCTRIG10 - ePWM3, ADCSOCA</p> <p>0Bh ADCTRIG11 - ePWM4, ADCSOCA</p> <p>0Ch ADCTRIG12 - ePWM4, ADCSOCA</p> <p>0Dh ADCTRIG13 - ePWM5, ADCSOCA</p> <p>0Eh ADCTRIG14 - ePWM5, ADCSOCA</p> <p>0Fh ADCTRIG15 - ePWM6, ADCSOCA</p> <p>10h ADCTRIG16 - ePWM6, ADCSOCA</p> <p>11h ADCTRIG17 - ePWM7, ADCSOCA</p> <p>12h ADCTRIG18 - ePWM7, ADCSOCA</p> <p>13h ADCTRIG19 - ePWM8, ADCSOCA</p> <p>14h ADCTRIG20 - ePWM8, ADCSOCA</p> <p>15h ADCTRIG21 - ePWM9, ADCSOCA</p> <p>16h ADCTRIG22 - ePWM9, ADCSOCA</p> <p>17h ADCTRIG23 - ePWM10, ADCSOCA</p> <p>18h ADCTRIG24 - ePWM10, ADCSOCA</p> <p>19h ADCTRIG25 - ePWM11, ADCSOCA</p> <p>1Ah ADCTRIG26 - ePWM11, ADCSOCA</p> <p>1Bh ADCTRIG27 - ePWM12, ADCSOCA</p> <p>1Ch ADCTRIG28 - ePWM12, ADCSOCA</p> <p>1Dh ADCTRIG29 - CPU2 Timer 0, TINT0n</p> <p>1Eh ADCTRIG30 - CPU2 Timer 1, TINT1n</p> <p>1Fh ADCTRIG31 - CPU2 Timer 2, TINT2n</p>
19	RESERVED	R	0h	Reserved

Table 9-44. ADCSOC15CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-15	CHSEL	R/W	0h	<p>SOC Channel Select. Selects the channel to be converted when this SOC is received by the ADC.</p> <p>Single-ended Signaling Mode (SIGNALMODE = 0):</p> <p>0h ADCIN0</p> <p>1h ADCIN1</p> <p>2h ADCIN2</p> <p>3h ADCIN3</p> <p>4h ADCIN4</p> <p>5h ADCIN5</p> <p>6h ADCIN6</p> <p>7h ADCIN7</p> <p>8h ADCIN8</p> <p>9h ADCIN9</p> <p>Ah ADCIN10</p> <p>Bh ADCIN11</p> <p>Ch ADCIN12</p> <p>Dh ADCIN13</p> <p>Eh ADCIN14</p> <p>Fh ADCIN15</p> <p>Differential Signaling Mode (SIGNALMODE = 1):</p> <p>0h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>1h ADCIN0 (non-inverting) and ADCIN1 (inverting)</p> <p>2h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>3h ADCIN2 (non-inverting) and ADCIN3 (inverting)</p> <p>4h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>5h ADCIN4 (non-inverting) and ADCIN5 (inverting)</p> <p>6h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>7h ADCIN6 (non-inverting) and ADCIN7 (inverting)</p> <p>8h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>9h ADCIN8 (non-inverting) and ADCIN9 (inverting)</p> <p>Ah ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Bh ADCIN10 (non-inverting) and ADCIN11 (inverting)</p> <p>Ch ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Dh ADCIN12 (non-inverting) and ADCIN13 (inverting)</p> <p>Eh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p> <p>Fh ADCIN14 (non-inverting) and ADCIN15 (inverting)</p>
14-9	RESERVED	R	0h	Reserved
8-0	ACQPS	R/W	0h	<p>SOC Acquisition Prescale. Controls the sample and hold window for this SOC. The minimum acquisition time is one ADCCLK cycle.</p> <p>000h Sample window is 1 system clock cycle wide</p> <p>001h Sample window is 2 system clock cycles wide</p> <p>002h Sample window is 3 system clock cycles wide</p> <p>...</p> <p>1FFh Sample window is 512 system clock cycles wide</p>

9.4.2.33 ADCEVTSTAT Register (Offset = 30h) [reset = 0h]

ADCEVTSTAT is shown in [Figure 9-48](#) and described in [Table 9-45](#).

ADC Event Status Register

Figure 9-48. ADCEVTSTAT Register

15	14	13	12	11	10	9	8
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI
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
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-45. 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.
13	PPB4TRIPLO	R	0h	Post Processing Block 4 Trip Low Flag. When set indicates a digital compare trip low event has occurred.
12	PPB4TRIPHI	R	0h	Post Processing Block 4 Trip High Flag. When set indicates a digital compare trip high event has occurred.
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.
9	PPB3TRIPLO	R	0h	Post Processing Block 3 Trip Low Flag. When set indicates a digital compare trip low event has occurred.
8	PPB3TRIPHI	R	0h	Post Processing Block 3 Trip High Flag. When set indicates a digital compare trip high event has occurred.
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.
5	PPB2TRIPLO	R	0h	Post Processing Block 2 Trip Low Flag. When set indicates a digital compare trip low event has occurred.
4	PPB2TRIPHI	R	0h	Post Processing Block 2 Trip High Flag. When set indicates a digital compare trip high event has occurred.
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.
1	PPB1TRIPLO	R	0h	Post Processing Block 1 Trip Low Flag. When set indicates a digital compare trip low event has occurred.
0	PPB1TRIPHI	R	0h	Post Processing Block 1 Trip High Flag. When set indicates a digital compare trip high event has occurred.

9.4.2.34 ADCEVTCLR Register (Offset = 32h) [reset = 0h]

ADCEVTCLR is shown in [Figure 9-49](#) and described in [Table 9-46](#).

ADC Event Clear Register

Figure 9-49. ADCEVTCLR 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-46. 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.
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.
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.
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.
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.
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.
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.
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.

9.4.2.35 ADCEVTSEL Register (Offset = 34h) [reset = 0h]

ADCEVTSEL is shown in [Figure 9-50](#) and described in [Table 9-47](#).

ADC Event Selection Register

Figure 9-50. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-47. 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.
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.
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.
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.
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.
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.
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.
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.
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.
3	RESERVED	R	0h	Reserved

Table 9-47. ADCEVTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.
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.

9.4.2.36 ADCEVTINTSEL Register (Offset = 36h) [reset = 0h]

ADCEVTINTSEL is shown in [Figure 9-51](#) and described in [Table 9-48](#).

ADC Event Interrupt Selection Register

Figure 9-51. ADCEVTINTSEL 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-48. 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.
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.
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.
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.
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.
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.
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.
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.
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.
3	RESERVED	R	0h	Reserved

Table 9-48. ADCEVTINTSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.
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.

9.4.2.37 ADCCOUNTER Register (Offset = 39h) [reset = 0h]

ADCCOUNTER is shown in [Figure 9-52](#) and described in [Table 9-49](#).

ADC Counter Register

Figure 9-52. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-49. 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.

9.4.2.38 ADCREV Register (Offset = 3Ah) [reset = X]

ADCREV is shown in [Figure 9-53](#) and described in [Table 9-50](#).

ADC Revision Register

Figure 9-53. ADCREV Register

15	14	13	12	11	10	9	8
REV							
R-X							
7	6	5	4	3	2	1	0
TYPE							
R-4h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-50. ADCREV Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	REV	R	X	ADC Revision. To allow documentation of differences between revisions. First version is labeled as 00h.
7-0	TYPE	R	4h	ADC Type. Always set to 4 for this ADC.

9.4.2.39 ADCOFFTRIM Register (Offset = 3Bh) [reset = 0h]

ADCOFFTRIM is shown in [Figure 9-54](#) and described in [Table 9-51](#).

ADC Offset Trim Register

Figure 9-54. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-51. ADCOFFTRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	OFFTRIM	R/W	0h	ADC Offset Trim. Adjusts the conversion results of the converter up or down to account for offset error in the ADC. A different offset trim is required for each combination of resolution and signal mode. Using the <code>AdcSetMode</code> function to set the resolution and signal mode will ensure that the correct offset trim is loaded. Range is +127 steps to -128 steps (2's compliment format). Regardless of the converter resolution, the size of each trim step is $(VREFHI-VREFLO)/65536$.

9.4.2.40 ADCPPB1CONFIG Register (Offset = 40h) [reset = 0h]

ADCPPB1CONFIG is shown in [Figure 9-55](#) and described in [Table 9-52](#).

ADC PPB1 Config Register

Figure 9-55. ADCPPB1CONFIG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			TWOSCOMPE N	CONFIG			
R-0h			R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-52. ADCPPB1CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 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 appropriate ADCPPBxRESULT register. 0 ADCPPBxRESULT = ADCRESULTx - ADCSOCxOFFREF 1 ADCPPBxRESULT = ADCSOCxOFFREF - ADCRESULTx

Table 9-52. ADCPPB1CONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 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 this post processing block</p> <p>0001 SOC1/EOC1/RESULT1 is associated with this post processing block</p> <p>0010 SOC2/EOC2/RESULT2 is associated with this post processing block</p> <p>0011 SOC3/EOC3/RESULT3 is associated with this post processing block</p> <p>0100 SOC4/EOC4/RESULT4 is associated with this post processing block</p> <p>0101 SOC5/EOC5/RESULT5 is associated with this post processing block</p> <p>0110 SOC6/EOC6/RESULT6 is associated with this post processing block</p> <p>0111 SOC7/EOC7/RESULT7 is associated with this post processing block</p> <p>1000 SOC8/EOC8/RESULT8 is associated with this post processing block</p> <p>1001 SOC9/EOC9/RESULT9 is associated with this post processing block</p> <p>1010 SOC10/EOC10/RESULT10 is associated with this post processing block</p> <p>1011 SOC11/EOC11/RESULT11 is associated with this post processing block</p> <p>1100 SOC12/EOC12/RESULT12 is associated with this post processing block</p> <p>1101 SOC13/EOC13/RESULT13 is associated with this post processing block</p> <p>1110 SOC14/EOC14/RESULT14 is associated with this post processing block</p> <p>1111 SOC15/EOC15/RESULT15 is associated with this post processing block</p>

9.4.2.41 ADCPPB1STAMP Register (Offset = 41h) [reset = 0h]

ADCPPB1STAMP is shown in [Figure 9-56](#) and described in [Table 9-53](#).

ADC PPB1 Sample Delay Time Stamp Register

Figure 9-56. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-53. ADCPPB1STAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 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.

9.4.2.42 ADCPPB1OFFCAL Register (Offset = 42h) [reset = 0h]

ADCPPB1OFFCAL is shown in [Figure 9-57](#) and described in [Table 9-54](#).

ADC PPB1 Offset Calibration Register

Figure 9-57. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-54. 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 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 03FFh before being stored into the ADCRESULT register.</p>

9.4.2.43 ADCPPB1OFFREF Register (Offset = 43h) [reset = 0h]

ADCPPB1OFFREF is shown in [Figure 9-58](#) and described in [Table 9-55](#).

ADC PPB1 Offset Reference Register

Figure 9-58. ADCPPB1OFFREF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFREF															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-55. ADCPPB1OFFREF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 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.</p> <p>0001h ADCRESULT - 1 is passed on.</p> <p>0002h ADCRESULT - 2 is passed on.</p> <p>...</p> <p>8000h ADCRESULT - 32,768 is passed on.</p> <p>...</p> <p>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.</p>

9.4.2.44 ADCPPB1TRIPHI Register (Offset = 44h) [reset = 0h]

ADCPPB1TRIPHI is shown in [Figure 9-59](#) and described in [Table 9-56](#).

ADC PPB1 Trip High Register

Figure 9-59. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-56. 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.
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 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 associated ADCPPBxRESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRSULT bit field of the associated ADCPPBxRESULT register.

9.4.2.45 ADCPPB1TRIPLO Register (Offset = 46h) [reset = 0h]

ADCPPB1TRIPLO is shown in [Figure 9-60](#) and described in [Table 9-57](#).

ADC PPB1 Trip Low/Trigger Time Stamp Register

Figure 9-60. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-57. ADCPPB1TRIPLO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 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.
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.
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 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 associated ADCPPBxRESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the associated ADCPPBxRESULT register.

9.4.2.46 ADCPPB2CONFIG Register (Offset = 48h) [reset = 0h]

ADCPPB2CONFIG is shown in [Figure 9-61](#) and described in [Table 9-58](#).

ADC PPB2 Config Register

Figure 9-61. ADCPPB2CONFIG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			TWOSCOMPE N	CONFIG			
R-0h			R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-58. ADCPPB2CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	TWOSCOMPEN	R/W	0h	<p>ADC Post Processing Block 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 appropriate ADCPPBxRESULT register.</p> <p>0 ADCPPBxRESULT = ADCRESULTx - ADCSOCxOFFREF</p> <p>1 ADCPPBxRESULT = ADCSOCxOFFREF - ADCRESULTx</p>

Table 9-58. ADCPPB2CONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 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 this post processing block</p> <p>0001 SOC1/EOC1/RESULT1 is associated with this post processing block</p> <p>0010 SOC2/EOC2/RESULT2 is associated with this post processing block</p> <p>0011 SOC3/EOC3/RESULT3 is associated with this post processing block</p> <p>0100 SOC4/EOC4/RESULT4 is associated with this post processing block</p> <p>0101 SOC5/EOC5/RESULT5 is associated with this post processing block</p> <p>0110 SOC6/EOC6/RESULT6 is associated with this post processing block</p> <p>0111 SOC7/EOC7/RESULT7 is associated with this post processing block</p> <p>1000 SOC8/EOC8/RESULT8 is associated with this post processing block</p> <p>1001 SOC9/EOC9/RESULT9 is associated with this post processing block</p> <p>1010 SOC10/EOC10/RESULT10 is associated with this post processing block</p> <p>1011 SOC11/EOC11/RESULT11 is associated with this post processing block</p> <p>1100 SOC12/EOC12/RESULT12 is associated with this post processing block</p> <p>1101 SOC13/EOC13/RESULT13 is associated with this post processing block</p> <p>1110 SOC14/EOC14/RESULT14 is associated with this post processing block</p> <p>1111 SOC15/EOC15/RESULT15 is associated with this post processing block</p>

9.4.2.47 ADCPPB2STAMP Register (Offset = 49h) [reset = 0h]

ADCPPB2STAMP is shown in [Figure 9-62](#) and described in [Table 9-59](#).

ADC PPB2 Sample Delay Time Stamp Register

Figure 9-62. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-59. ADCPPB2STAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 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.

9.4.2.48 ADCPPB2OFFCAL Register (Offset = 4Ah) [reset = 0h]

ADCPPB2OFFCAL is shown in [Figure 9-63](#) and described in [Table 9-60](#).

ADC PPB2 Offset Calibration Register

Figure 9-63. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-60. 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 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 03FFh before being stored into the ADCRESULT register.</p>

9.4.2.49 ADCPPB2OFFREF Register (Offset = 4Bh) [reset = 0h]

ADCPPB2OFFREF is shown in [Figure 9-64](#) and described in [Table 9-61](#).

ADC PPB2 Offset Reference Register

Figure 9-64. ADCPPB2OFFREF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFREF															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-61. ADCPPB2OFFREF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 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.</p> <p>0001h ADCRESULT - 1 is passed on.</p> <p>0002h ADCRESULT - 2 is passed on.</p> <p>...</p> <p>8000h ADCRESULT - 32,768 is passed on.</p> <p>...</p> <p>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.</p>

9.4.2.50 ADCPPB2TRIPHI Register (Offset = 4Ch) [reset = 0h]

ADCPPB2TRIPHI is shown in [Figure 9-65](#) and described in [Table 9-62](#).

ADC PPB2 Trip High Register

Figure 9-65. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-62. 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.
15-0	LIMITHI	R/W	0h	ADC Post Processing Block Trip High Limit. This value sets the digital comparator trip high limit.

9.4.2.51 ADCPPB2TRIPLO Register (Offset = 4Eh) [reset = 0h]

ADCPPB2TRIPLO is shown in [Figure 9-66](#) and described in [Table 9-63](#).

ADC PPB2 Trip Low/Trigger Time Stamp Register

Figure 9-66. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-63. ADCPPB2TRIPLO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 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.
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.
15-0	LIMITLO	R/W	0h	ADC Post Processing Block Trip Low Limit. This value sets the digital comparator trip low limit.

9.4.2.52 ADCPPB3CONFIG Register (Offset = 50h) [reset = 0h]

ADCPPB3CONFIG is shown in [Figure 9-67](#) and described in [Table 9-64](#).

ADC PPB3 Config Register

Figure 9-67. ADCPPB3CONFIG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			TWOSCOMPE N	CONFIG			
R-0h			R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-64. ADCPPB3CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	TWOSCOMPEN	R/W	0h	<p>ADC Post Processing Block 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 appropriate ADCPPBxRESULT register.</p> <p>0 ADCPPBxRESULT = ADCRESULTx - ADCSOCxOFFREF</p> <p>1 ADCPPBxRESULT = ADCSOCxOFFREF - ADCRESULTx</p>

Table 9-64. ADCPPB3CONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 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 this post processing block</p> <p>0001 SOC1/EOC1/RESULT1 is associated with this post processing block</p> <p>0010 SOC2/EOC2/RESULT2 is associated with this post processing block</p> <p>0011 SOC3/EOC3/RESULT3 is associated with this post processing block</p> <p>0100 SOC4/EOC4/RESULT4 is associated with this post processing block</p> <p>0101 SOC5/EOC5/RESULT5 is associated with this post processing block</p> <p>0110 SOC6/EOC6/RESULT6 is associated with this post processing block</p> <p>0111 SOC7/EOC7/RESULT7 is associated with this post processing block</p> <p>1000 SOC8/EOC8/RESULT8 is associated with this post processing block</p> <p>1001 SOC9/EOC9/RESULT9 is associated with this post processing block</p> <p>1010 SOC10/EOC10/RESULT10 is associated with this post processing block</p> <p>1011 SOC11/EOC11/RESULT11 is associated with this post processing block</p> <p>1100 SOC12/EOC12/RESULT12 is associated with this post processing block</p> <p>1101 SOC13/EOC13/RESULT13 is associated with this post processing block</p> <p>1110 SOC14/EOC14/RESULT14 is associated with this post processing block</p> <p>1111 SOC15/EOC15/RESULT15 is associated with this post processing block</p>

9.4.2.53 ADCPPB3STAMP Register (Offset = 51h) [reset = 0h]

ADCPPB3STAMP is shown in [Figure 9-68](#) and described in [Table 9-65](#).

ADC PPB3 Sample Delay Time Stamp Register

Figure 9-68. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-65. ADCPPB3STAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 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.

9.4.2.54 ADCPPB3OFFCAL Register (Offset = 52h) [reset = 0h]

ADCPPB3OFFCAL is shown in [Figure 9-69](#) and described in [Table 9-66](#).

ADC PPB3 Offset Calibration Register

Figure 9-69. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-66. 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 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 03FFh before being stored into the ADCRESULT register.</p>

9.4.2.55 ADCPPB3OFFREF Register (Offset = 53h) [reset = 0h]

ADCPPB3OFFREF is shown in [Figure 9-70](#) and described in [Table 9-67](#).

ADC PPB3 Offset Reference Register

Figure 9-70. ADCPPB3OFFREF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFREF															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-67. ADCPPB3OFFREF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 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.</p> <p>0001h ADCRESULT - 1 is passed on.</p> <p>0002h ADCRESULT - 2 is passed on.</p> <p>...</p> <p>8000h ADCRESULT - 32,768 is passed on.</p> <p>...</p> <p>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.</p>

9.4.2.56 ADCPPB3TRIPHI Register (Offset = 54h) [reset = 0h]

ADCPPB3TRIPHI is shown in [Figure 9-71](#) and described in [Table 9-68](#).

ADC PPB3 Trip High Register

Figure 9-71. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-68. 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.
15-0	LIMITHI	R/W	0h	ADC Post Processing Block Trip High Limit. This value sets the digital comparator trip high limit.

9.4.2.57 ADCPPB3TRIPLO Register (Offset = 56h) [reset = 0h]

ADCPPB3TRIPLO is shown in [Figure 9-72](#) and described in [Table 9-69](#).

ADC PPB3 Trip Low/Trigger Time Stamp Register

Figure 9-72. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-69. ADCPPB3TRIPLO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 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.
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.
15-0	LIMITLO	R/W	0h	ADC Post Processing Block Trip Low Limit. This value sets the digital comparator trip low limit.

9.4.2.58 ADCPPB4CONFIG Register (Offset = 58h) [reset = 0h]

ADCPPB4CONFIG is shown in [Figure 9-73](#) and described in [Table 9-70](#).

ADC PPB4 Config Register

Figure 9-73. ADCPPB4CONFIG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			TWOSCOMPE N	CONFIG			
R-0h			R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-70. ADCPPB4CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	TWOSCOMPEN	R/W	0h	<p>ADC Post Processing Block 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 appropriate ADCPPBxRESULT register.</p> <p>0 ADCPPBxRESULT = ADCRESULTx - ADCSOCxOFFREF</p> <p>1 ADCPPBxRESULT = ADCSOCxOFFREF - ADCRESULTx</p>

Table 9-70. ADCPPB4CONFIG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 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 this post processing block</p> <p>0001 SOC1/EOC1/RESULT1 is associated with this post processing block</p> <p>0010 SOC2/EOC2/RESULT2 is associated with this post processing block</p> <p>0011 SOC3/EOC3/RESULT3 is associated with this post processing block</p> <p>0100 SOC4/EOC4/RESULT4 is associated with this post processing block</p> <p>0101 SOC5/EOC5/RESULT5 is associated with this post processing block</p> <p>0110 SOC6/EOC6/RESULT6 is associated with this post processing block</p> <p>0111 SOC7/EOC7/RESULT7 is associated with this post processing block</p> <p>1000 SOC8/EOC8/RESULT8 is associated with this post processing block</p> <p>1001 SOC9/EOC9/RESULT9 is associated with this post processing block</p> <p>1010 SOC10/EOC10/RESULT10 is associated with this post processing block</p> <p>1011 SOC11/EOC11/RESULT11 is associated with this post processing block</p> <p>1100 SOC12/EOC12/RESULT12 is associated with this post processing block</p> <p>1101 SOC13/EOC13/RESULT13 is associated with this post processing block</p> <p>1110 SOC14/EOC14/RESULT14 is associated with this post processing block</p> <p>1111 SOC15/EOC15/RESULT15 is associated with this post processing block</p>

9.4.2.59 ADCPPB4STAMP Register (Offset = 59h) [reset = 0h]

ADCPPB4STAMP is shown in [Figure 9-74](#) and described in [Table 9-71](#).

ADC PPB4 Sample Delay Time Stamp Register

Figure 9-74. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-71. ADCPPB4STAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 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.

9.4.2.60 ADCPPB4OFFCAL Register (Offset = 5Ah) [reset = 0h]

ADCPPB4OFFCAL is shown in [Figure 9-75](#) and described in [Table 9-72](#).

ADC PPB4 Offset Calibration Register

Figure 9-75. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-72. 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 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 03FFh before being stored into the ADCRESULT register.</p>

9.4.2.61 ADCPPB4OFFREF Register (Offset = 5Bh) [reset = 0h]

ADCPPB4OFFREF is shown in [Figure 9-76](#) and described in [Table 9-73](#).

ADC PPB4 Offset Reference Register

Figure 9-76. ADCPPB4OFFREF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFREF															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-73. ADCPPB4OFFREF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 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.</p> <p>0001h ADCRESULT - 1 is passed on.</p> <p>0002h ADCRESULT - 2 is passed on.</p> <p>...</p> <p>8000h ADCRESULT - 32,768 is passed on.</p> <p>...</p> <p>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.</p>

9.4.2.62 ADCPPB4TRIPHI Register (Offset = 5Ch) [reset = 0h]

ADCPPB4TRIPHI is shown in [Figure 9-77](#) and described in [Table 9-74](#).

ADC PPB4 Trip High Register

Figure 9-77. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-74. 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.
15-0	LIMITHI	R/W	0h	ADC Post Processing Block Trip High Limit. This value sets the digital comparator trip high limit.

9.4.2.63 ADCPPB4TRIPLO Register (Offset = 5Eh) [reset = 0h]

ADCPPB4TRIPLO is shown in [Figure 9-78](#) and described in [Table 9-75](#).

ADC PPB4 Trip Low/Trigger Time Stamp Register

Figure 9-78. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-75. ADCPPB4TRIPLO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 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.
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.
15-0	LIMITLO	R/W	0h	ADC Post Processing Block Trip Low Limit. This value sets the digital comparator trip low limit.

9.4.2.64 ADCINLTRIM1 Register (Offset = 70h) [reset = 0h]

ADCINLTRIM1 is shown in [Figure 9-79](#) and described in [Table 9-76](#).

ADC Linearity Trim 1 Register

Figure 9-79. 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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-76. ADCINLTRIM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM31TO0	R/W	0h	ADC Linearity Trim Bits 31-0. 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.

9.4.2.65 ADCINLTRIM2 Register (Offset = 72h) [reset = 0h]

ADCINLTRIM2 is shown in [Figure 9-80](#) and described in [Table 9-77](#).

ADC Linearity Trim 2 Register

Figure 9-80. 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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-77. ADCINLTRIM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM63TO32	R/W	0h	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.

9.4.2.66 ADCINLTRIM3 Register (Offset = 74h) [reset = 0h]

ADCINLTRIM3 is shown in [Figure 9-81](#) and described in [Table 9-78](#).

ADC Linearity Trim 3 Register

Figure 9-81. 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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-78. ADCINLTRIM3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM95TO64	R/W	0h	ADC Linearity Trim Bits 95-64. 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.

9.4.2.67 ADCINLTRIM4 Register (Offset = 76h) [reset = 0h]

ADCINLTRIM4 is shown in [Figure 9-82](#) and described in [Table 9-79](#).

ADC Linearity Trim 4 Register

Figure 9-82. ADCINLTRIM4 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INLTRIM127TO96																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-79. ADCINLTRIM4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM127TO96	R/W	0h	ADC Linearity Trim Bits 127-96. 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.

9.4.2.68 ADCINLTRIM5 Register (Offset = 78h) [reset = 0h]

ADCINLTRIM5 is shown in [Figure 9-83](#) and described in [Table 9-80](#).

ADC Linearity Trim 5 Register

Figure 9-83. ADCINLTRIM5 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INLTRIM159TO128																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-80. ADCINLTRIM5 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM159TO128	R/W	0h	ADC Linearity Trim Bits 159-128. 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.

9.4.2.69 ADCINLTRIM6 Register (Offset = 7Ah) [reset = 0h]

ADCINLTRIM6 is shown in [Figure 9-84](#) and described in [Table 9-81](#).

ADC Linearity Trim 6 Register

Figure 9-84. ADCINLTRIM6 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INLTRIM191TO160																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-81. ADCINLTRIM6 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INLTRIM191TO160	R/W	0h	ADC Linearity Trim Bits 191-160. 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.

9.4.3 ADC_RESULT_REGS Registers

Table 9-82 lists the memory-mapped registers for the ADC_RESULT_REGS. All register offset addresses not listed in Table 9-82 should be considered as reserved locations and the register contents should not be modified.

Table 9-82. ADC_RESULT_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCRESULT0	ADC Result 0 Register		Go
1h	ADCRESULT1	ADC Result 1 Register		Go
2h	ADCRESULT2	ADC Result 2 Register		Go
3h	ADCRESULT3	ADC Result 3 Register		Go
4h	ADCRESULT4	ADC Result 4 Register		Go
5h	ADCRESULT5	ADC Result 5 Register		Go
6h	ADCRESULT6	ADC Result 6 Register		Go
7h	ADCRESULT7	ADC Result 7 Register		Go
8h	ADCRESULT8	ADC Result 8 Register		Go
9h	ADCRESULT9	ADC Result 9 Register		Go
Ah	ADCRESULT10	ADC Result 10 Register		Go
Bh	ADCRESULT11	ADC Result 11 Register		Go
Ch	ADCRESULT12	ADC Result 12 Register		Go
Dh	ADCRESULT13	ADC Result 13 Register		Go
Eh	ADCRESULT14	ADC Result 14 Register		Go
Fh	ADCRESULT15	ADC Result 15 Register		Go
10h	ADCPPB1RESULT	ADC Post Processing Block 1 Result Register		Go
12h	ADCPPB2RESULT	ADC Post Processing Block 2 Result Register		Go
14h	ADCPPB3RESULT	ADC Post Processing Block 3 Result Register		Go
16h	ADCPPB4RESULT	ADC Post Processing Block 4 Result Register		Go

9.4.3.1 ADCRESULT0 Register (Offset = 0h) [reset = 0h]

ADCRESULT0 is shown in [Figure 9-85](#) and described in [Table 9-83](#).

ADC Result 0 Register

Figure 9-85. ADCRESULT0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-83. ADCRESULT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC0, the digital result is placed in this bit field.

9.4.3.2 ADCRESULT1 Register (Offset = 1h) [reset = 0h]

ADCRESULT1 is shown in [Figure 9-86](#) and described in [Table 9-84](#).

ADC Result 1 Register

Figure 9-86. ADCRESULT1 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-84. ADCRESULT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC1, the digital result is placed in this bit field.

9.4.3.3 ADCRESULT2 Register (Offset = 2h) [reset = 0h]

ADCRESULT2 is shown in [Figure 9-87](#) and described in [Table 9-85](#).

ADC Result 2 Register

Figure 9-87. ADCRESULT2 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-85. ADCRESULT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC2, the digital result is placed in this bit field.

9.4.3.4 ADCRESULT3 Register (Offset = 3h) [reset = 0h]

ADCRESULT3 is shown in [Figure 9-88](#) and described in [Table 9-86](#).

ADC Result 3 Register

Figure 9-88. ADCRESULT3 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-86. ADCRESULT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC3, the digital result is placed in this bit field.

9.4.3.5 ADCRESULT4 Register (Offset = 4h) [reset = 0h]

ADCRESULT4 is shown in [Figure 9-89](#) and described in [Table 9-87](#).

ADC Result 4 Register

Figure 9-89. ADCRESULT4 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-87. ADCRESULT4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC4, the digital result is placed in this bit field.

9.4.3.6 ADCRESULT5 Register (Offset = 5h) [reset = 0h]

ADCRESULT5 is shown in [Figure 9-90](#) and described in [Table 9-88](#).

ADC Result 5 Register

Figure 9-90. ADCRESULT5 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-88. ADCRESULT5 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC5, the digital result is placed in this bit field.

9.4.3.7 ADCRESULT6 Register (Offset = 6h) [reset = 0h]

ADCRESULT6 is shown in [Figure 9-91](#) and described in [Table 9-89](#).

ADC Result 6 Register

Figure 9-91. ADCRESULT6 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-89. ADCRESULT6 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC6, the digital result is placed in this bit field.

9.4.3.8 ADCRESULT7 Register (Offset = 7h) [reset = 0h]

ADCRESULT7 is shown in [Figure 9-92](#) and described in [Table 9-90](#).

ADC Result 7 Register

Figure 9-92. ADCRESULT7 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-90. ADCRESULT7 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC7, the digital result is placed in this bit field.

9.4.3.9 ADCRESULT8 Register (Offset = 8h) [reset = 0h]

ADCRESULT8 is shown in [Figure 9-93](#) and described in [Table 9-91](#).

ADC Result 8 Register

Figure 9-93. ADCRESULT8 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-91. ADCRESULT8 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC8, the digital result is placed in this bit field.

9.4.3.10 ADCRESULT9 Register (Offset = 9h) [reset = 0h]

ADCRESULT9 is shown in [Figure 9-94](#) and described in [Table 9-92](#).

ADC Result 9 Register

Figure 9-94. ADCRESULT9 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-92. ADCRESULT9 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC9, the digital result is placed in this bit field.

9.4.3.11 ADCRESULT10 Register (Offset = Ah) [reset = 0h]

ADCRESULT10 is shown in [Figure 9-95](#) and described in [Table 9-93](#).

ADC Result 10 Register

Figure 9-95. ADCRESULT10 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-93. ADCRESULT10 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC10, the digital result is placed in this bit field.

9.4.3.12 ADCRESULT11 Register (Offset = Bh) [reset = 0h]

ADCRESULT11 is shown in [Figure 9-96](#) and described in [Table 9-94](#).

ADC Result 11 Register

Figure 9-96. ADCRESULT11 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-94. ADCRESULT11 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC11, the digital result is placed in this bit field.

9.4.3.13 ADCRESULT12 Register (Offset = Ch) [reset = 0h]

ADCRESULT12 is shown in [Figure 9-97](#) and described in [Table 9-95](#).

ADC Result 12 Register

Figure 9-97. ADCRESULT12 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-95. ADCRESULT12 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC12, the digital result is placed in this bit field.

9.4.3.14 ADCRESULT13 Register (Offset = Dh) [reset = 0h]

ADCRESULT13 is shown in [Figure 9-98](#) and described in [Table 9-96](#).

ADC Result 13 Register

Figure 9-98. ADCRESULT13 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-96. ADCRESULT13 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC13, the digital result is placed in this bit field.

9.4.3.15 ADCRESULT14 Register (Offset = Eh) [reset = 0h]

ADCRESULT14 is shown in [Figure 9-99](#) and described in [Table 9-97](#).

ADC Result 14 Register

Figure 9-99. ADCRESULT14 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-97. ADCRESULT14 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC14, the digital result is placed in this bit field.

9.4.3.16 ADCRESULT15 Register (Offset = Fh) [reset = 0h]

ADCRESULT15 is shown in [Figure 9-100](#) and described in [Table 9-98](#).

ADC Result 15 Register

Figure 9-100. ADCRESULT15 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-98. ADCRESULT15 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result. 16-bit ADC result. After the ADC completes a conversion of SOC15, the digital result is placed in this bit field.

9.4.3.17 ADCPPB1RESULT Register (Offset = 10h) [reset = 0h]

ADCPPB1RESULT is shown in [Figure 9-101](#) and described in [Table 9-99](#).

ADC Post Processing Block 1 Result Register

Figure 9-101. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-99. 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 13, and reflect the same value as bit 12.
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result. The result of the offset/reference subtraction post conversion processing is stored 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.

9.4.3.18 ADCPPB2RESULT Register (Offset = 12h) [reset = 0h]

ADCPPB2RESULT is shown in [Figure 9-102](#) and described in [Table 9-100](#).

ADC Post Processing Block 2 Result Register

Figure 9-102. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-100. ADCPPB2RESULT 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 13, and reflect the same value as bit 12.
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result. The result of the offset/reference subtraction post conversion processing is stored 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.

9.4.3.19 ADCPPB3RESULT Register (Offset = 14h) [reset = 0h]

ADCPPB3RESULT is shown in [Figure 9-103](#) and described in [Table 9-101](#).

ADC Post Processing Block 3 Result Register

Figure 9-103. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-101. 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 13, and reflect the same value as bit 12.
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result. The result of the offset/reference subtraction post conversion processing is stored 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.

9.4.3.20 ADCPPB4RESULT Register (Offset = 16h) [reset = 0h]

ADCPPB4RESULT is shown in [Figure 9-104](#) and described in [Table 9-102](#).

ADC Post Processing Block 4 Result Register

Figure 9-104. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-102. ADCPPB4RESULT 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 13, and reflect the same value as bit 12.
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result. The result of the offset/reference subtraction post conversion processing is stored 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.

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.

Topic	Page
10.1 Buffered Digital to Analog Converter (DAC) Overview	1374
10.2 Using the DAC	1374
10.3 Lock Registers	1375
10.4 Registers	1375

10.1 Buffered Digital to Analog Converter (DAC) Overview

The buffered DAC module consists of an internal reference DAC and an analog output buffer that is capable of driving an external load. An integrated pull-down resistor on the DAC output helps to provide a known pin voltage when the output buffer is disabled. This pull-down resistor cannot be disabled and remains as a passive component on the pin, even for other shared pinmux functions. Software writes to the DAC value register can take effect immediately or can be synchronized with PWMSYNC events.

10.1.1 Features

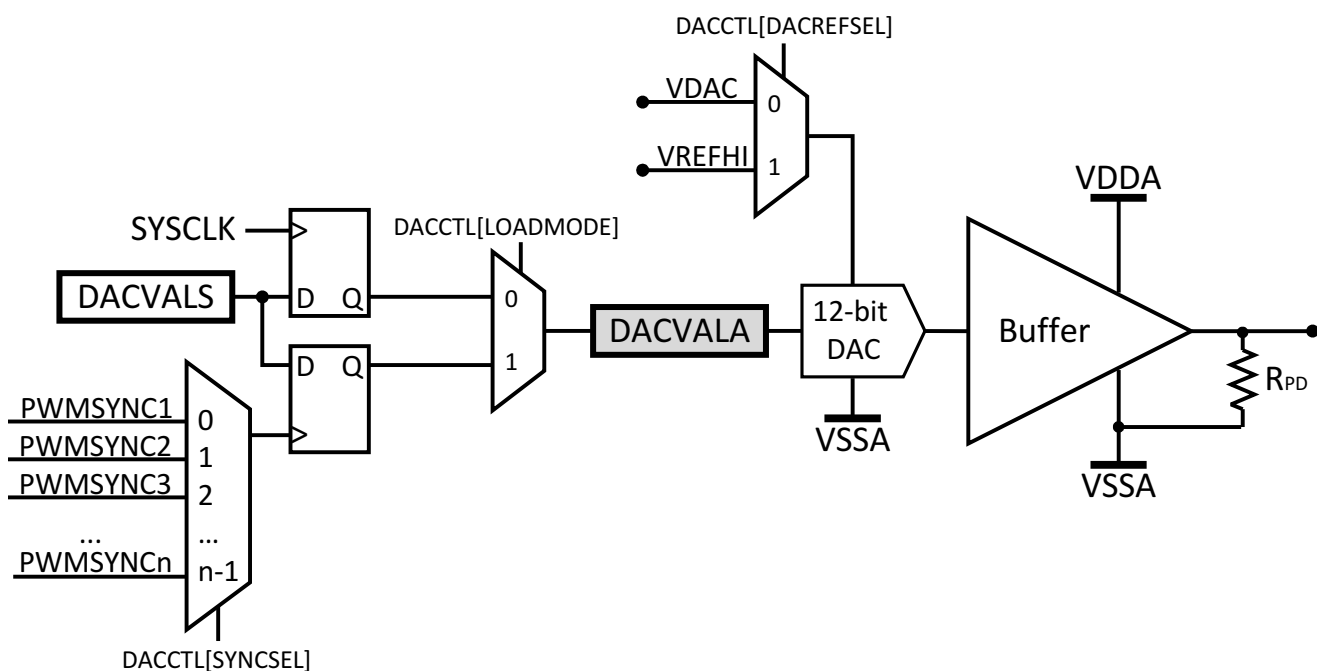
Each buffered DAC has the following features:

- 12-bit programmable internal DAC
- Selectable reference voltage
- Pull-down resistor on output
- Ability to synchronize with PWMSYNC

10.1.2 Block Diagram

The block diagram for the buffered DAC is shown in [Figure 10-1](#).

Figure 10-1. DAC Module Block Diagram



10.2 Using the DAC

The reference voltage for the internal DAC is selectable between the **VDAC** and **VREFHI**.

Two sets of **DACVAL** registers are present in the buffered DAC module: **DACVALA** and **DACVALS**. **DACVALA** is a read-only register that actively controls the DAC value. **DACVALS** is a writable shadow register that loads into **DACVALA** either immediately or synchronized with the next **PWMSYNC** event.

The ideal output of the internal DAC can be calculated as follows:

$$V_{DAC} = \frac{DACVALA * DACREF}{4096}$$

The Buffered DAC output buffer may exhibit non-linear behavior near the supply rails (**VDDA/VSSA**). See the device datasheet to determine the valid operating range of the Buffered DAC.

10.3 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.

10.4 Registers

10.4.1 DAC Base Addresses

Table 10-1. 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
DaccRegs	DAC_REGS	0x0000_5C20	0x0000_5C2F

10.4.2 DAC_REGS Registers

Table 10-2 lists the memory-mapped registers for the DAC_REGS. All register offset addresses not listed in Table 10-2 should be considered as reserved locations and the register contents should not be modified.

Table 10-2. DAC_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	DACREV	DAC Revision Register		Go
1h	DACCTL	DAC Control Register	EALLOW	Go
2h	DACVALA	DAC Value Register - Active		Go
3h	DACVALS	DAC Value Register - Shadow		Go
4h	DACOUTEN	DAC Output Enable Register	EALLOW	Go
5h	DACLOCK	DAC Lock Register	EALLOW	Go
6h	DACTRIM	DAC Trim Register	EALLOW	Go

10.4.2.1 DACREV Register (Offset = 0h) [reset = 0h]

DACREV is shown in [Figure 10-2](#) and described in [Table 10-3](#).

DAC Revision Register

Figure 10-2. DACREV Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
REV							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-3. DACREV Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	REV	R	0h	DAC Revision

10.4.2.2 DACCTL Register (Offset = 1h) [reset = 0h]

DACCTL is shown in [Figure 10-3](#) and described in [Table 10-4](#).

DAC Control Register

Figure 10-3. DACCTL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SYNCSEL				RESERVED	LOADMODE	RESERVED	DACREFSEL
R/W-0h				R-0h	R/W-0h	R-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-4. DACCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-4	SYNCSEL	R/W	0h	DAC PWMSYNC select. Determines which PWMSYNC signal will update the DACVALA register. Where n represents the maximum number of PWMSYNC signals available on the device: 0 PWMSYNC1 1 PWMSYNC2 2 PWMSYNC3 ... n-1 PWMSYNcn
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 PWMSYNC specified by SYNCSEL
1	RESERVED	R	0h	Reserved
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/VREFLO are the reference voltages

10.4.2.3 DACVALA Register (Offset = 2h) [reset = 0h]

DACVALA is shown in [Figure 10-4](#) and described in [Table 10-5](#).

DAC Value Register - Active

Figure 10-4. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-5. 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

10.4.2.4 DACVALS Register (Offset = 3h) [reset = 0h]

DACVALS is shown in [Figure 10-5](#) and described in [Table 10-6](#).

DAC Value Register - Shadow

Figure 10-5. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-6. 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

10.4.2.5 DACOUTEN Register (Offset = 4h) [reset = 0h]

DACOUTEN is shown in [Figure 10-6](#) and described in [Table 10-7](#).

DAC Output Enable Register

Figure 10-6. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-7. 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

10.4.2.6 DACLOCK Register (Offset = 5h) [reset = 0h]

DACLOCK is shown in [Figure 10-7](#) and described in [Table 10-8](#).

DAC Lock Register

Figure 10-7. DACLOCK Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					DACOUTEN	DACVAL	DACCTL
R-0h					R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-8. DACLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	DACOUTEN	R/SOnce	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.
1	DACVAL	R/SOnce	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.
0	DACCTL	R/SOnce	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.

10.4.2.7 DACTRIM Register (Offset = 6h) [reset = 0h]

DACTRIM is shown in [Figure 10-8](#) and described in [Table 10-9](#).

DAC Trim Register

Figure 10-8. DACTRIM Register

15	14	13	12	11	10	9	8
RESERVED				RESERVED			
R-0h				R-0h			
7	6	5	4	3	2	1	0
OFFSET_TRIM							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 10-9. DACTRIM Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R	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.

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, and voltage trip monitoring.

Topic	Page
11.1 CMPSS Overview	1385
11.2 Comparator	1386
11.3 Internal DAC	1386
11.4 Ramp Generator	1387
11.5 Digital Filter	1388
11.6 Registers	1390

11.1 CMPSS Overview

The comparator subsystem is built around a number of modules, each comprising a pair of analog comparators. Comparators are denoted "H" or "L" within each module. 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 always driven from an external pin, but the negative input can be driven by either an external pin or by an internal programmable 12-bit DAC. Each comparator output passes through a programmable digital filter that can remove spurious trip signals. A ramp generator circuit is optionally available to control the internal DAC value for one comparator in the subsystem.

11.1.1 Features

CMPSS Features

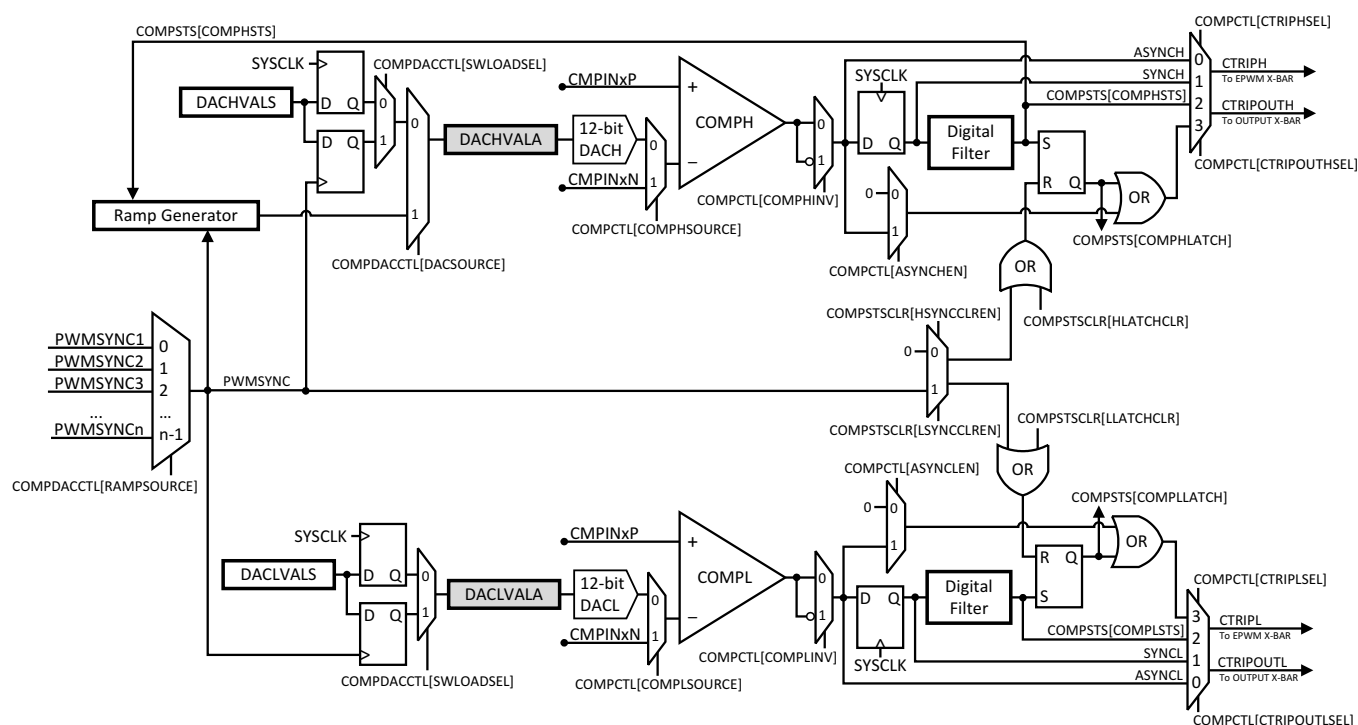
- Each CMPSS module includes:
 - Two analog comparators
 - Two programmable 12-bit DACs
 - One ramp generator
 - Two digital filters
- Ability to synchronize with PWMSYNC
- Option for negative input of comparator to be driven by an external signal or by an internal DAC
- Option to choose between VDDA or VDACC to be the DAC reference voltage

11.1.2 Block Diagram

The block diagram for the CMPSS is shown in [Figure 11-1](#).

- 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.
- CTRIPxOUT 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.

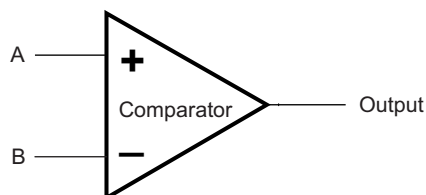
Figure 11-1. CMPSS Module Block Diagram



11.2 Comparator

The comparator generates a digital output which indicates whether the voltage on the positive input is greater than the voltage on the negative input.

Figure 11-2. Comparator Digital Output



Voltages	Output
Voltage A > Voltage B	1
Voltage A < Voltage B	0

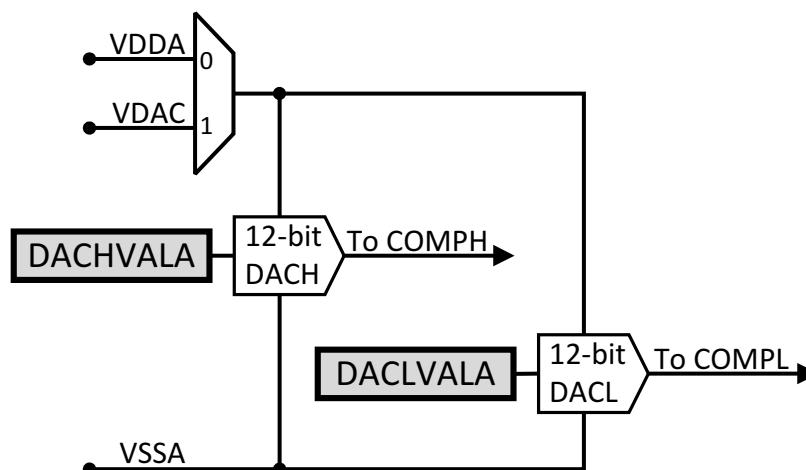
11.3 Internal DAC

Each 12-bit internal DAC can be configured to drive the reference voltage into the negative input of its respective comparator. The DAC output is internal only and cannot be observed externally.

Two sets of DACxVAL registers are present for each DAC: DACxVALA and DACxVALS. DACxVALA is a read-only register that actively controls the DAC value. DACxVALS is a writable shadow register that loads into DACxVALA either immediately or synchronized with the next PWMSYNC event. The high DAC (DACH) can optionally source its DACHVALA value from the Ramp Generator instead of DACHVALS.

The operating range of the DACs is bounded by its high and low reference voltages. The high voltage reference is VDDA by default, but it can be configured to be VDAC if desired.

Figure 11-3. DAC Reference Select

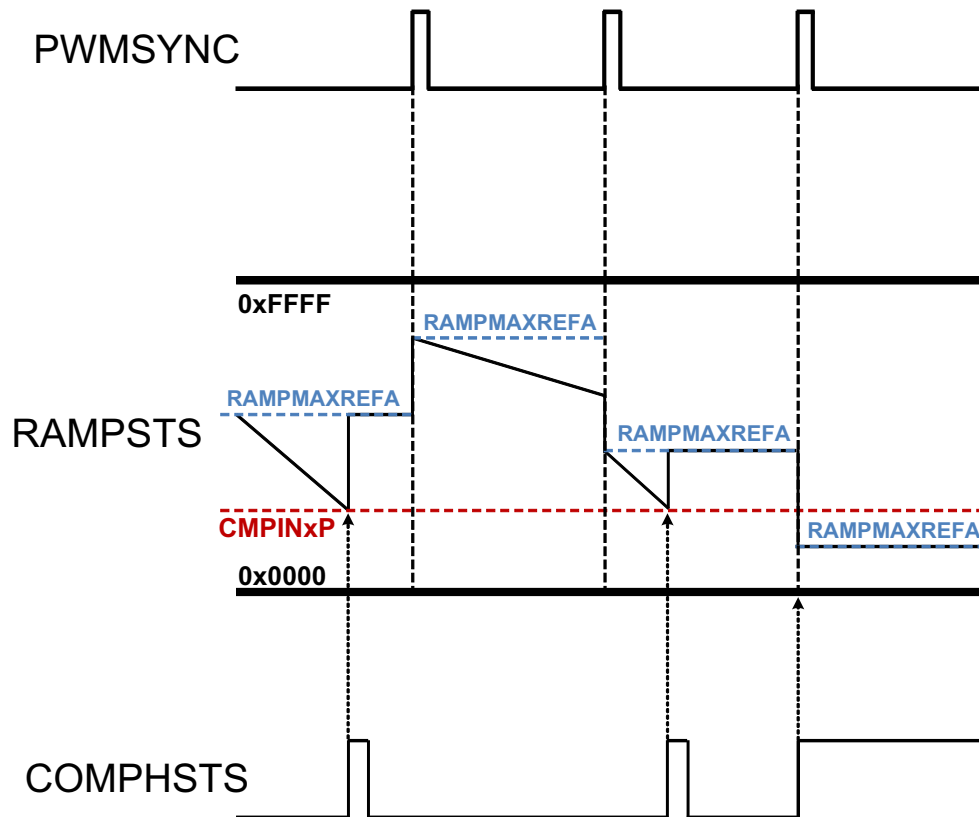


The ideal output voltage of the DACs can be calculated as follows:

Figure 11-4. Output Voltage Calculation

$$V_{DACx} = \frac{DACxVALA * DACREF}{4096}$$

Figure 11-6. Ramp Generator Behavior



11.5 Digital Filter

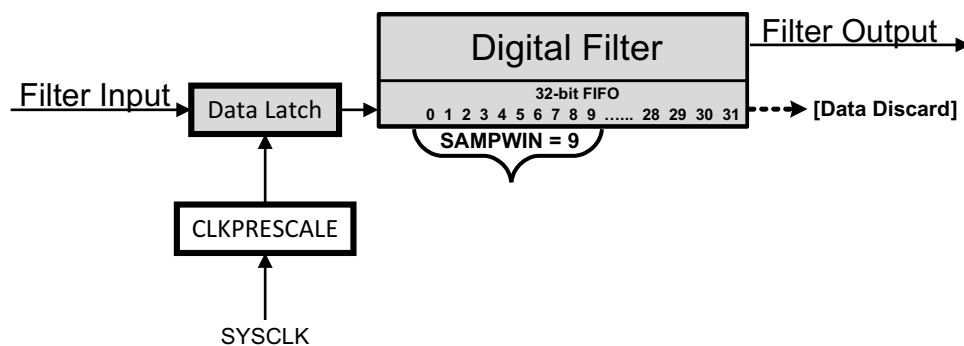
The digital filter works on a window of FIFO samples ($\text{SAMPWIN} + 1$) taken from the input. The filter output resolves to the majority value of the sample window, where majority is defined by the threshold (THRESH) value. If the majority threshold is not satisfied, the filter output remains unchanged.

For proper operation, the value of THRESH must be greater than $\text{SAMPWIN} / 2$.

A prescale function (CLKPRESCALE) determines the filter sampling rate, where the filter FIFO captures one sample every CLKPRESCALE system clocks. Old data from the FIFO is discarded.

A conceptual model of the digital filter is shown in Figure 11-7.

Figure 11-7. Digital Filter Behavior



Equivalent C code of the filter implementation is shown below:

```
if (FILTER_OUTPUT == 0) {
    if (Num_ls_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 1;
    }
}
```

```
    }  
  }  
  else {  
    if (Num_0s_in_SAMPWIN >= THRESH) {  
      FILTER_OUTPUT = 0;  
    }  
  }  
}
```

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 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. Configure the CTRIP and CTRIPOUT signal paths
5. If desired, configure the ePWM and GPIO modules to accept the filtered signals

11.6 Registers

11.6.1 CMPSS Base Addresses

Table 11-1. 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
Cmpss8Regs	CMPSS_REGS	0x0000_5D60	0x0000_5D7F

11.6.2 CMPSS_REGS Registers

Table 11-2 lists the memory-mapped registers for the CMPSS_REGS. All register offset addresses not listed in Table 11-2 should be considered as reserved locations and the register contents should not be modified.

Table 11-2. CMPSS_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	COMPCTL	CMPSS Comparator Control Register	EALLOW	Go
1h	COMPHYSCTL	CMPSS Comparator Hysteresis Control Register	EALLOW	Go
2h	COMPSTS	CMPSS Comparator Status Register		Go
3h	COMPSTSLR	CMPSS Comparator Status Clear Register	EALLOW	Go
4h	COMPDACCTL	CMPSS DAC Control Register	EALLOW	Go
6h	DACHVALS	CMPSS High DAC Value Shadow Register		Go
7h	DACHVALA	CMPSS High DAC Value Active Register		Go
8h	RAMPMAXREFA	CMPSS Ramp Max Reference Active Register		Go
Ah	RAMPMAXREFS	CMPSS Ramp Max Reference Shadow Register		Go
Ch	RAMPDECVALA	CMPSS Ramp Decrement Value Active Register		Go
Eh	RAMPDECVALS	CMPSS Ramp Decrement Value Shadow Register		Go
10h	RAMPSTS	CMPSS Ramp Status Register		Go
12h	DACLVALS	CMPSS Low DAC Value Shadow Register		Go
13h	DACLVALA	CMPSS Low DAC Value Active Register		Go
14h	RAMPDLYA	CMPSS Ramp Delay Active Register		Go
15h	RAMPDLYS	CMPSS Ramp Delay Shadow Register		Go
16h	CTRIPLFILCTL	CTRIPL Filter Control Register	EALLOW	Go
17h	CTRIPLFILCLKCTL	CTRIPL Filter Clock Control Register	EALLOW	Go
18h	CTRIPHFILCTL	CTRIPH Filter Control Register	EALLOW	Go
19h	CTRIPHFILCLKCTL	CTRIPH Filter Clock Control Register	EALLOW	Go
1Ah	COMPLOCK	CMPSS Lock Register	EALLOW	Go

11.6.2.1 COMPCTL Register (Offset = 0h) [reset = 0h]

COMPCTL is shown in [Figure 11-8](#) and described in [Table 11-3](#).

CMPSS Comparator Control Register

Figure 11-8. COMPCTL Register

15	14	13	12	11	10	9	8
COMPDAE	ASYNCLN	CTRIPOUTLSEL	CTRIPLSEL	COMPLINV	COMPLSOURCE		
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
RESERVED	ASYNCHEN	CTRIPOUTHSEL	CTRIPHSEL	COMPHINV	COMPHSOURCE		
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-3. COMPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	COMPDAE	R/W	0h	Comparator/DAC enable. 0 Comparator/DAC disabled 1 Comparator/DAC enabled
14	ASYNCLN	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
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
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
9	COMPLINV	R/W	0h	Low comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted
8	COMPLSOURCE	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
7	RESERVED	R	0h	Reserved
6	ASYNCHEN	R/W	0h	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. 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

Table 11-3. COMPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5-4	CTRIPOUTHSEL	R/W	0h	High comparator CTRIPOUTH source select. 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
3-2	CTRIPHSEL	R/W	0h	High comparator CTRIPH source select. 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
1	COMPHINV	R/W	0h	High comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted
0	COMPHSOURCE	R/W	0h	High comparator input source. 0 Inverting input of comparator driven by internal DAC 1 Inverting input of comparator driven through external pin

11.6.2.2 COMPHYCTL Register (Offset = 1h) [reset = 0h]

COMPHYCTL is shown in [Figure 11-9](#) and described in [Table 11-4](#).

CMPSS Comparator Hysteresis Control Register

Figure 11-9. COMPHYCTL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					COMPHYS		
R-0h					R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-4. COMPHYCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2-0	COMPHYS	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

11.6.2.3 COMPSTS Register (Offset = 2h) [reset = 0h]

COMPSTS is shown in [Figure 11-10](#) and described in [Table 11-5](#).

CMPSS Comparator Status Register

Figure 11-10. 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	COMPHSTS
R-0h						R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-5. 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
8	COMPLSTS	R	0h	Low comparator digital filter output
7-2	RESERVED	R	0h	Reserved
1	COMPHLATCH	R	0h	Latched value of high comparator digital filter output
0	COMPHSTS	R	0h	High comparator digital filter output

11.6.2.4 COMPSTSCLR Register (Offset = 3h) [reset = 0h]

COMPSTSCLR is shown in [Figure 11-11](#) and described in [Table 11-6](#).

CMPSS Comparator Status Clear Register

Figure 11-11. COMPSTSCLR Register

15	14	13	12	11	10	9	8
RESERVED					LSYNCCLREN	LLATCHCLR	RESERVED
R-0h					R/W-0h	R=0/W=1-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED					HSYNCCLREN	HLATCHCLR	RESERVED
R-0h					R/W-0h	R=0/W=1-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-6. COMPSTSCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	LSYNCCLREN	R/W	0h	Low comparator latch PWMSYNC clear. Enable PWMSYNC reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. 0 PWMSYNC will not reset latch 1 PWMSYNC will reset latch
9	LLATCHCLR	R=0/W=1	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]
8-3	RESERVED	R	0h	Reserved
2	HSYNCCLREN	R/W	0h	High comparator latch PWMSYNC clear. Enable PWMSYNC reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. 0 PWMSYNC will not reset latch 1 PWMSYNC will reset latch
1	HLATCHCLR	R=0/W=1	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]
0	RESERVED	R	0h	Reserved

11.6.2.5 COMPDACCTL Register (Offset = 4h) [reset = 0h]

COMPDACCTL is shown in [Figure 11-12](#) and described in [Table 11-7](#).

CMPSS DAC Control Register

Figure 11-12. COMPDACCTL Register

15	14	13	12	11	10	9	8
FREESOFT		RESERVED					
R/W-0h		R-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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-7. 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 PWMSYNC during emulation suspend 1Xb Ramp generator runs freely
13-8	RESERVED	R	0h	Reserved
7	SWLOADSEL	R/W	0h	Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or PWMSYNC. 0 DACxVALA is updated from DACxVALS on SYSCLK 1 DACxVALA is updated from DACxVALS on PWMSYNC
6	RAMPLOADSEL	R/W	0h	Ramp load select. Determines whether RAMPSTS is updated from RAMPMAXREFA or RAMPMAXREFS when COMPSTS[COMPHSTS] is triggered. 0 RAMPSTS is loaded from RAMPMAXREFA 1 RAMPSTS is loaded from RAMPMAXREFS
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
4-1	RAMPSOURCE	R/W	0h	Ramp generator source select. Determines which PWMSYNcx signal is used within the CMPSS module. Where n represents the maximum number of PWMSYNC signals available on the device: 0 PWMSYNc1 1 PWMSYNc2 2 PWMSYNc3 ... n-1 PWMSYNcn
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

11.6.2.6 DACHVALS Register (Offset = 6h) [reset = 0h]

DACHVALS is shown in [Figure 11-13](#) and described in [Table 11-8](#).

CMPSS High DAC Value Shadow Register

Figure 11-13. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-8. 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].

11.6.2.7 DACHVALA Register (Offset = 7h) [reset = 0h]

DACHVALA is shown in [Figure 11-14](#) and described in [Table 11-9](#).

CMPSS High DAC Value Active Register

Figure 11-14. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-9. 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.

11.6.2.8 RAMPMAXREFA Register (Offset = 8h) [reset = 0h]

RAMPMAXREFA is shown in [Figure 11-15](#) and described in [Table 11-10](#).

CMPSS Ramp Max Reference Active Register

Figure 11-15. RAMPMAXREFA Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMPMAXREF															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-10. 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.

11.6.2.9 RAMPMAXREFS Register (Offset = Ah) [reset = 0h]

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

CMPSS Ramp Max Reference Shadow Register

Figure 11-16. RAMPMAXREFS Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMPMAXREF															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-11. 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.

11.6.2.10 RAMPDECVALA Register (Offset = Ch) [reset = 0h]

RAMPDECVALA is shown in [Figure 11-17](#) and described in [Table 11-12](#).

CMPSS Ramp Decrement Value Active Register

Figure 11-17. RAMPDECVALA Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMPDECVAL															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-12. 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.

11.6.2.11 RAMPDECVALS Register (Offset = Eh) [reset = 0h]

RAMPDECVALS is shown in [Figure 11-18](#) and described in [Table 11-13](#).

CMPSS Ramp Decrement Value Shadow Register

Figure 11-18. RAMPDECVALS Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMPDECVAL															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-13. 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.

11.6.2.12 RAMPSTS Register (Offset = 10h) [reset = 0h]

RAMPSTS is shown in [Figure 11-19](#) and described in [Table 11-14](#).

CMPSS Ramp Status Register

Figure 11-19. RAMPSTS Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAMPVALUE															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-14. RAMPSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	RAMPVALUE	R	0h	Ramp value. Present value of ramp generator.

11.6.2.13 DACLVALS Register (Offset = 12h) [reset = 0h]

DACLVALS is shown in [Figure 11-20](#) and described in [Table 11-15](#).

CMPSS Low DAC Value Shadow Register

Figure 11-20. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-15. 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].

11.6.2.14 DACLVALA Register (Offset = 13h) [reset = 0h]

DACLVALA is shown in [Figure 11-21](#) and described in [Table 11-16](#).

CMPSS Low DAC Value Active Register

Figure 11-21. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-16. 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.

11.6.2.15 RAMPDLYA Register (Offset = 14h) [reset = 0h]

RAMPDLYA is shown in [Figure 11-22](#) and described in [Table 11-17](#).

CMPSS Ramp Delay Active Register

Figure 11-22. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-17. 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 decrements after a PWMSYNC is received.

11.6.2.16 RAMPDLYS Register (Offset = 15h) [reset = 0h]

RAMPDLYS is shown in [Figure 11-23](#) and described in [Table 11-18](#).

CMPSS Ramp Delay Shadow Register

Figure 11-23. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-18. 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.

11.6.2.17 CTRIPLFILCTL Register (Offset = 16h) [reset = 0h]

CTRIPLFILCTL is shown in [Figure 11-24](#) and described in [Table 11-19](#).

CTRIPL Filter Control Register

Figure 11-24. CTRIPLFILCTL Register

15	14	13	12	11	10	9	8
FILINIT	RESERVED			THRESH			SAMPWIN
R=0/W=1-0h	R-0h			R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
		SAMPWIN				RESERVED	
		R/W-0h				R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-19. CTRIPLFILCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	FILINIT	R=0/W=1	0h	Low filter initialization. 0 No effect 1 Initialize all samples to the filter input value
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.
8-4	SAMPWIN	R/W	0h	Low filter sample window size. Number of samples to monitor is SAMPWIN+1.
3-0	RESERVED	R	0h	Reserved

11.6.2.18 CTRIPLFILCLKCTL Register (Offset = 17h) [reset = 0h]

CTRIPLFILCLKCTL is shown in [Figure 11-25](#) and described in [Table 11-20](#).

CTRIPL Filter Clock Control Register

Figure 11-25. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-20. 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.

11.6.2.19 CTRIPHFILCTL Register (Offset = 18h) [reset = 0h]

CTRIPHFILCTL is shown in [Figure 11-26](#) and described in [Table 11-21](#).

CTRIPH Filter Control Register

Figure 11-26. CTRIPHFILCTL Register

15	14	13	12	11	10	9	8
FILINIT	RESERVED			THRESH			SAMPWIN
R=0/W=1-0h	R-0h			R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
		SAMPWIN				RESERVED	
		R/W-0h				R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-21. CTRIPHFILCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	FILINIT	R=0/W=1	0h	High filter initialization. 0 No effect 1 Initialize all samples to the filter input value
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.
8-4	SAMPWIN	R/W	0h	High filter sample window size. Number of samples to monitor is SAMPWIN+1.
3-0	RESERVED	R	0h	Reserved

11.6.2.20 CTRIPHFILCLKCTL Register (Offset = 19h) [reset = 0h]

CTRI PHFILCLKCTL is shown in [Figure 11-27](#) and described in [Table 11-22](#).

CTRI PH Filter Clock Control Register

Figure 11-27. CTRIPHFILCLKCTL 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-22. CTRIPHFILCLKCTL 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.

11.6.2.21 COMPLOCK Register (Offset = 1Ah) [reset = 0h]

COMPLOCK is shown in [Figure 11-28](#) and described in [Table 11-23](#).

CMPSS Lock Register

Figure 11-28. 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-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 11-23. COMPLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	CTRIIP	R/SOnce	0h	Lock write-access to the CTRIIPxFILCTL and CTRIIPxFILCLKCTL registers. 0 CTRIIPxFILCTL and CTRIIPxFILCLKCTL registers are not locked. Write 0 to this bit has no effect. 1 CTRIIPxFILCTL and CTRIIPxFILCLKCTL registers are locked. Only a system reset can clear this bit.
2	DACCTL	R/SOnce	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.
1	COMPHYSCTL	R/SOnce	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.
0	COMPCTL	R/SOnce	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.

Sigma Delta Filter Module (SDFM)

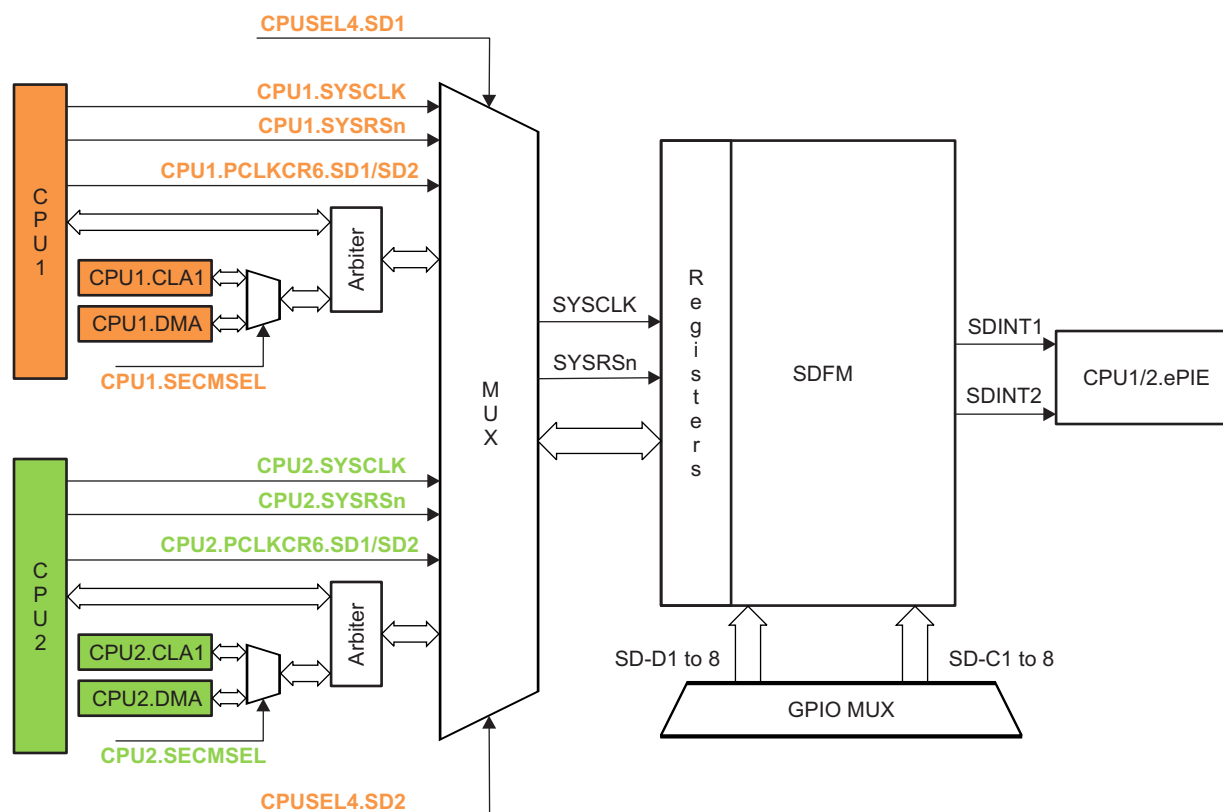
This chapter describes the sigma delta filter module (SDFM). SDFM is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each channel can receive an independent delta-sigma ($\Delta\Sigma$) modulator bit stream. The bit streams are processed by four individually-programmable digital decimation filters. The filter set includes a fast comparator for immediate digital threshold comparisons for over-current and under-current monitoring.

Topic	Page
12.1 SDFM Module Overview	1415
12.2 Configuring Device Pins	1418
12.3 Input Control Unit	1419
12.4 Comparator Unit	1421
12.5 Data Filter Unit	1422
12.6 Interrupt Unit	1427
12.7 Register Descriptions	1429
12.8 Registers	1430

12.1 SDFM Module Overview

Figure 12-1 shows the SDFM CPU interfaces:

Figure 12-1. Sigma Delta Filter Module (SDFM) CPU Interface



12.1.1 SDFM Features

The SDFM features include:

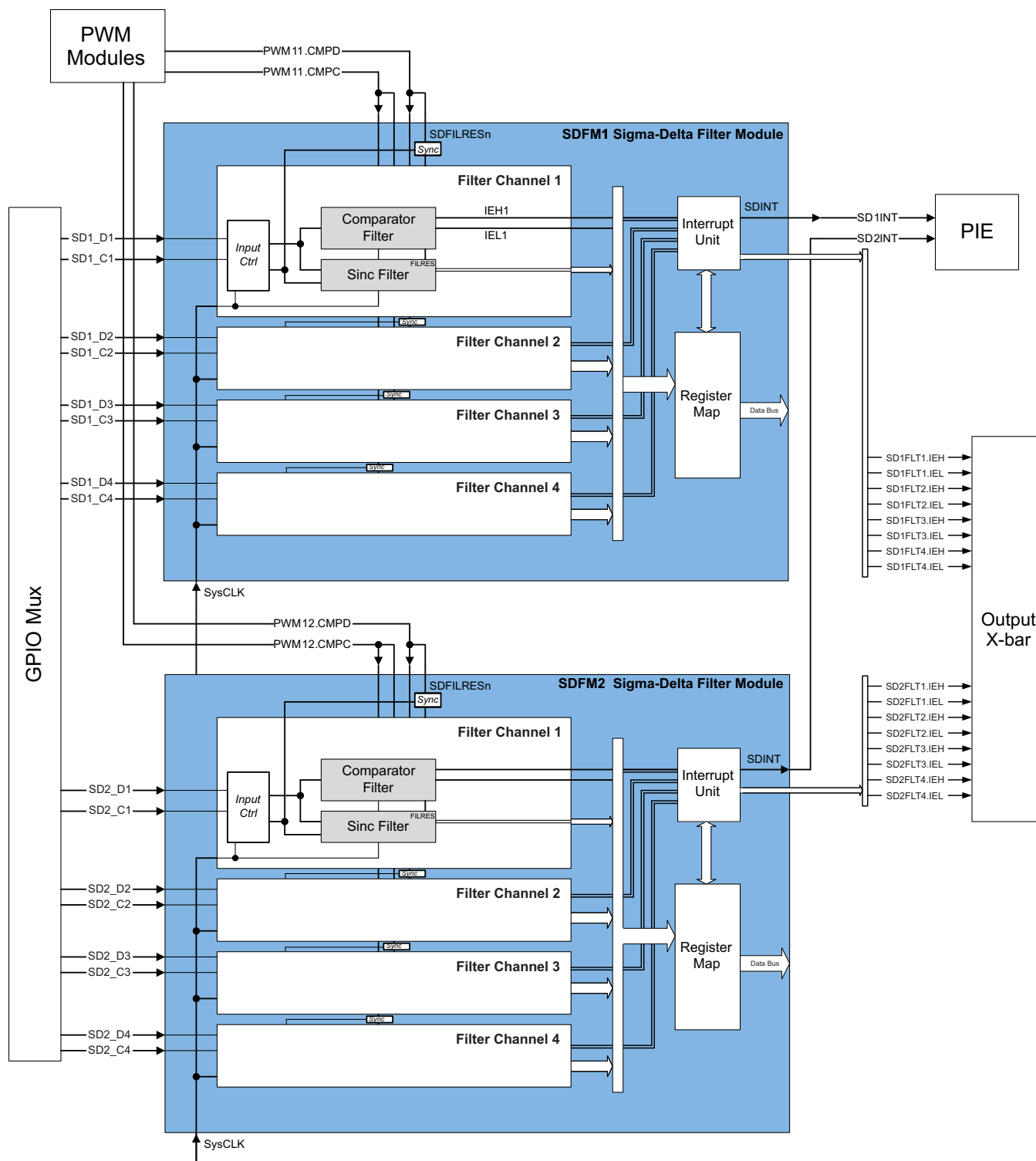
- 8 external pins per SDFM module
 - 4 sigma delta data input pins per SDFM module (SD-Dx, where x = 1 to 4)
 - 4 sigma delta clock input pins per SDFM module (SD-Cx, where x = 1 to 4)
- 4 different configurable modulator clock modes:
 - Mode 0: Modulator clock rate equals modulator data rate
 - Mode 1: Modulator clock rate running at half the modulator data rate
 - Mode 2: Modulator data is Manchester encoded. Modulator clock not required.
 - Mode 3: Modulator clock rate is double that of modulator data rate
- 4 independent configurable comparator units per SDFM module:
 - 4 different filter type selection (Sinc1/Sinc2/Sincfast/Sinc3) options available
 - Ability to detect over and under value conditions
 - OSR value for comparator programmable from 1 to 32
- 4 independent configurable sinc filter units per SDFM module:
 - 4 different filter type selection (Sinc1/Sinc2/Sincfast/Sinc3) options available
 - OSR value for filter unit programmable from 1 to 256
 - Ability to enable / disable individual filter module
 - Ability to synchronize all the 4 independent filters of a SDFM module using Master Filter Enable (MFE) bit (or) using PWM signals.

- Data filter output can be represented in either 16 bits (or) 32 bits
- PWMs can be used to generate a modulator clock for sigma delta modulators

12.1.2 Block Diagram

[Figure 12-2](#) shows the SDFM module block diagram. The SDFM port operation is configured and controlled by the registers listed in [Table 12-1](#).

Figure 12-2. Sigma Delta Filter Module (SDFM) 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
- Data filter unit
- Comparator unit

Figure 12-3 shows the block diagram of one filter module. When the SDDFPARMx.FILRESEN bit is set, the filter reset signal (FILRES) from the PWM module can be used to synchronize filters. It should be noted that the FILRES input is ONLY connected to the main Data Filter Unit and NOT the comparator, and that the reset function is limited to resetting the OSR counter. The FILRES input does not reset the data registers in the Filter Unit. Figure 12-4 shows how the PWM signals are connected to sigma delta module. In this device, PWM11 can be used to reset SDFM1 module and PWM12 can be used to reset SDFM2 module.

Figure 12-3. Block Diagram of One Filter Module

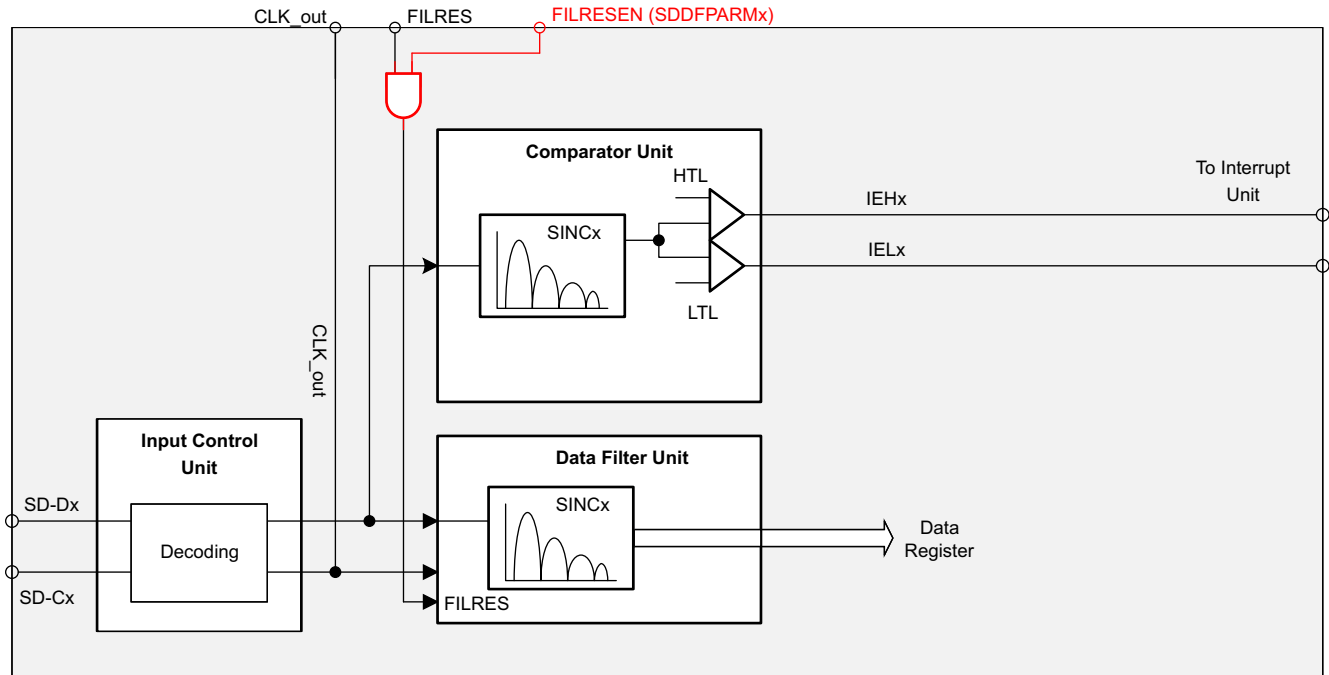
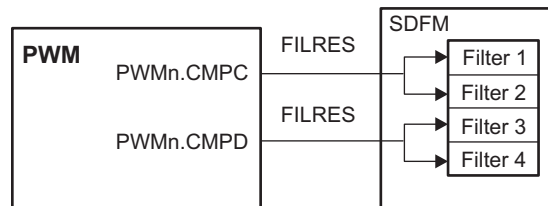


Figure 12-4. Typical PWM Interface to Sigma Delta Filter Module



Note: When using PWM11 (or) PWM12 for SDFM filter synchronization, users MUST ensure that ONLY ONE CMPC (or) CMPD event will be generated per PWM timer period. Using PWM in up-count (or) down-count mode would automatically ensure that you get ONLY one PWM (or) PWM event. But, if the user wishes to use up-down count mode, then they need to make sure that only one CMPC (or) CMPD event per PWM cycle is generated otherwise filter synchronizer will corrupt SDFM timing by providing two pulses per PWM cycle.

12.2 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.

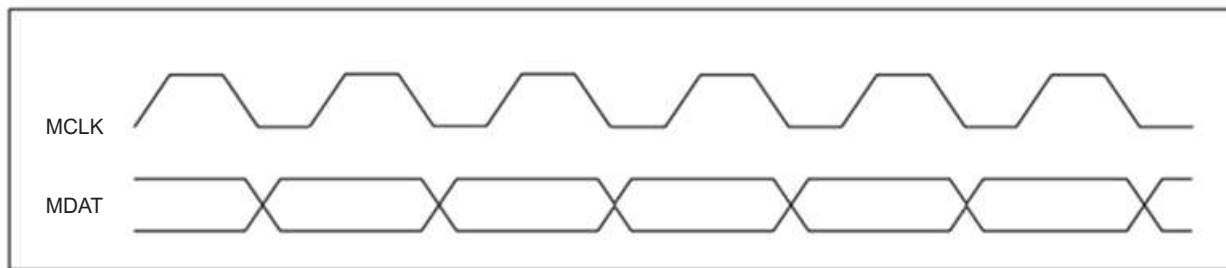
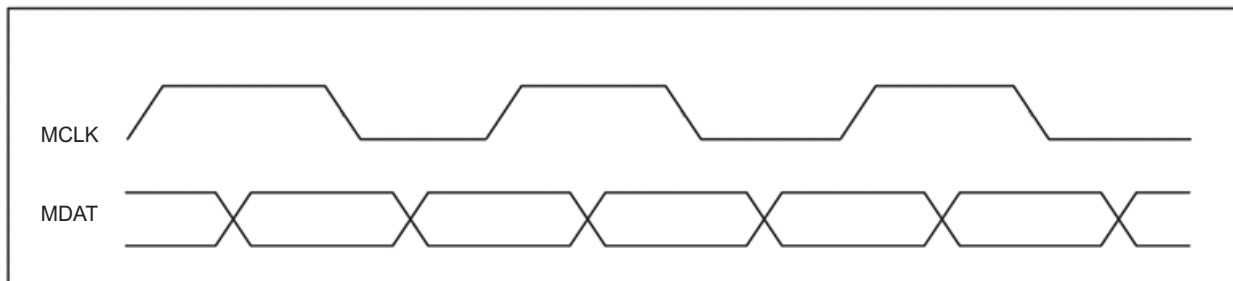
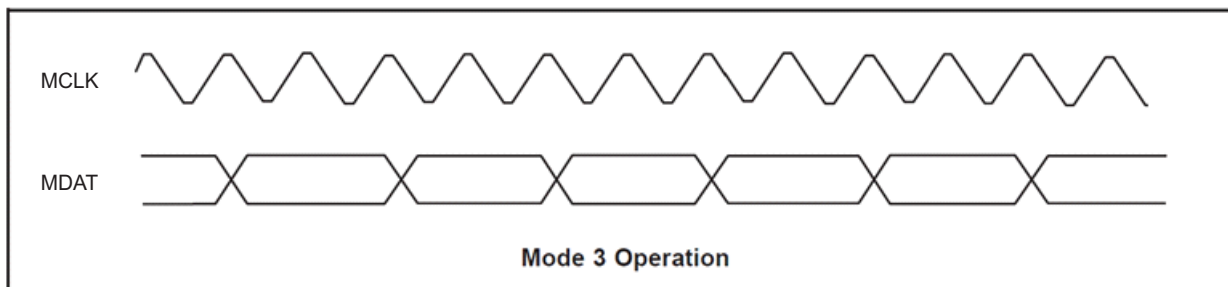
See the *GPIO* chapter for more details on GPIO mux and settings.

12.3 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 onto the data filter unit and comparator unit. This unit can be configured to receive the modulated data in four different modes. [Table 12-1](#) shows how SDCTLPARMx.MOD bits can be configured in these four different modes.

Table 12-1. 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 of the modulator data rate. The modulator data is strobed at every other positive modulator clock edge.


Mode 0 Operation

Mode 1 Operation

Mode 3 Operation

When MOD=2, data and modulated clock signals are encoded into modulated data as shown in [Figure 12-5](#). In this mode, the clock input SDCLKx.pin should be left floating. The input control unit performs continuous automatic calibration to achieve optimum decoding performance. The status of this calibration can be checked in the SDCTLPARMn register bits MS[10:0] and in the SDSTATUS register bits Manchester Locked (MALn) and Manchester Fail (MAFn).

12.3.1 Manchester Decoding

Manchester signaling is a method of encoding a data signal in such a way that it can be retrieved without the need of a separate clock line. When configured in Mode 2, the SDFM can translate a Manchester-encoded signal on SD-Dx into a clock signal and a data signal. An automatic calibration is continuously performed to optimize the decoding of the data.

The calibration mechanism follows this sequence:

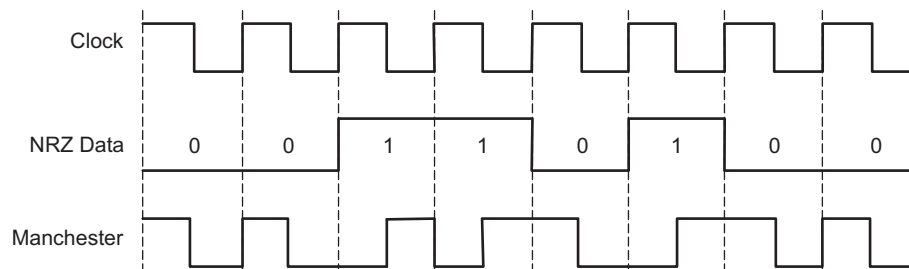
1. The modulator data is sampled at the frequency of the system clock (SYSCLK). The number of CLK cycles between transitions is counted and recorded for 1024 consecutive transitions.
2. The resulting array will have a '1' in the bit location that corresponds to the number of SYSCLK cycles counted between transitions. For example, the sequence shown in Table 12-2 means that there was at least one instance where three and four, as well as seven and eight, CLK cycles occurred between two transitions. This array is stored in the bits MS10–MS0 in the SDCTLPARMn Control Parameter register.
3. An algorithm looks for a group of zeros that has ones before and after it. If this pattern is not found, the bits MALx and MAFx in the SDSTATUS Register are set high.
4. If the algorithm is successful, it will use the location of the first '0' as the number of SYSCLK cycles needed to determine the frequency and which transitions are valid in the Manchester code.
5. The algorithm starts over from Step 2 automatically.

Table 12-2. Example Control Parameter Register

Value	0	0	0	1	1	0	0	1	1	0	0
Bit	MS10	MS9	MS8	MS7	MS6	MS5	MS4	MS3	MS2	MS1	MS0
CLK Cycles	11	10	9	8	7	6	5	4	3	2	1

Note: Bit refers to MS[] bits, not absolute SDCTLPARMn[] bit locations.

Figure 12-5. Manchester Coding Scheme



The MALx bit shows the status of the previous Manchester decoder calibration cycle. If it is high, the decoder calibration has failed on the previous calibration cycle. The MAFx bit shows if any failures have occurred since the last read of the SDSTATUS Register. Any MALx failure will cause MAFx to go high. MAFx is reset to low when the SDStatus Register is read.

The decoding procedure is performed continuously when the SDFM is configured for Modulator Mode 2 (Manchester encoded data). Note that the SYSCLK frequency must be at least six times the Manchester data rate for the decoder to perform properly.

12.4 Comparator Unit

An independent comparator unit allows the user to monitor input conditions with a fast settling time without sacrificing input measurement resolution. The comparator filter is a configurable sinc filter which supports the following filter types:- Sinc1, Sinc2, Sinc3 and Sincfast filter. Comparator filter OSR (COSR) value can be configured from 1 to 32. With Sinc3 as filter type and OSR = 32, a maximum 15-bit output width of 32,768 can be achieved. The output of the filter is compared with two programmed threshold levels to

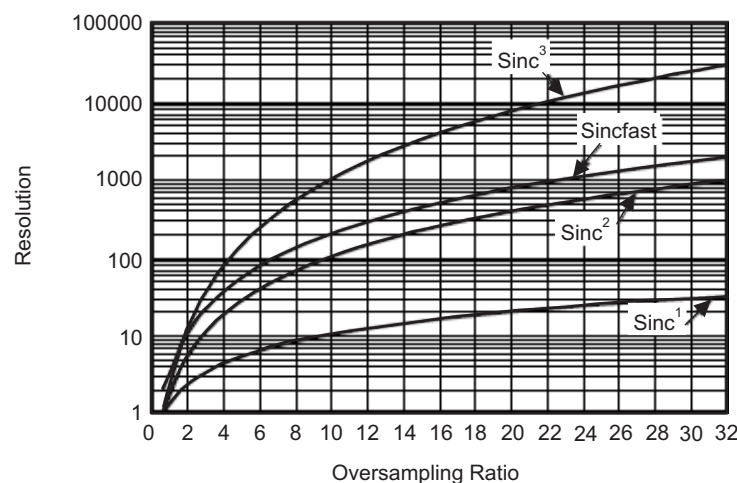
detect over- and under-value conditions. But output of the comparator filter cannot be read out of the filter. The comparator unit can be configured by high- and low-level Threshold registers (SDCMPHx and SDCMPLx) for each individual filter module. When an over- or under-value condition occurs, appropriate bits in SDIFLG (Interrupt Flag) register are set and then this register can be polled to see which condition caused interrupt signal.

The comparator filter unit and the data filter unit differ in the way they handle input data. The comparator filter unit translates a low input signal to a '0' and a high input signal to a '1', whereas the data filter unit uses '-1' and '1'. The resulting calculations give only positive values for the output of the comparator filter. The data representation is straight binary. Table 12-3 and Figure 12-6 show the different full-scale values that the comparator filter can store using different oversampling ratios.

Table 12-3. Peak Data Values for Different OSR/Filter Combinations

OSR	Sinc1	Sinc2	Sinc3	Sincfast
x	0 to x	0 to x ²	0 to x ³	0 to 2x ²
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

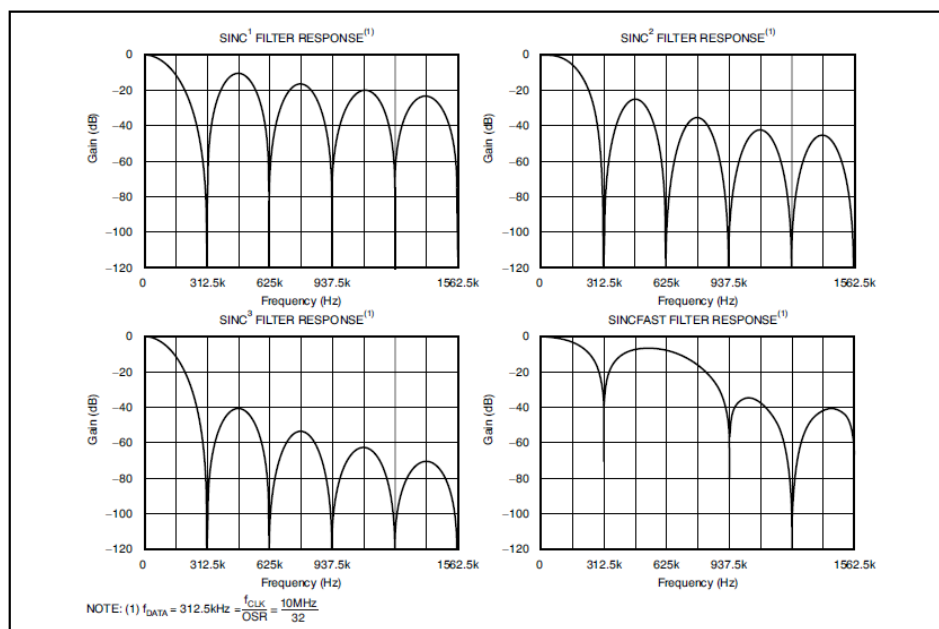
Figure 12-6. Comparator Filter Resolution



In order to achieve the maximum value, the delta-sigma modulator is operated at absolute maximum positive or negative full-scale, which is outside of the recommended full-scale range of 80% of most delta-sigma modulators.

12.5 Data Filter Unit

The data filter is a configurable sinc filter which supports the following filter types: Sinc1, Sinc2, Sinc3 and Sincfast filter. The data filter OSR (DOSR) value can be configured from 1 to 256. Figure 12-7 illustrates the frequency response of each type of filter when DOSR = 32 and modulator rate = 10 MHz.

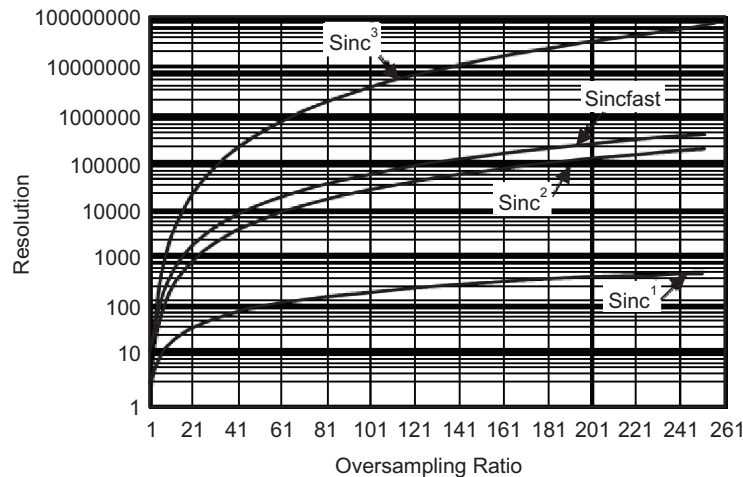
Figure 12-7. Frequency Response of Various Sinc Filters


The user can achieve a higher resolution by increasing the data filter oversampling ratio (DOSR) but this would reduce the data rate of the data filter. [Table 12-4](#) shows the peak data for different filter type or DOSR combinations and [Figure 12-8](#) shows the relationship between the DOSR and filter resolution. The highest resolution for any given DOSR can be achieved by using the Sinc3 filter and the highest possible resolution can be achieved with DOSR = 256 and Sinc3 filter.

Table 12-4. Peak Data Values for Different DOSR/Filter Combinations

DOSR	Sinc1	Sinc2	Sinc3	Sincfast
x	x	x^2	x^3	$2x^2$
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 16,777,215	-131,072 to 131,072

Figure 12-8. Data Filter Resolution



The data filter uses 25 bits to represent signed integer in two's complement format. The maximum possible resolution gives a 25-bit word (-16,777,216 to +16,777,215). Note that this value is only reached if the delta-sigma modulator is operated at absolute maximum positive or negative full-scale, which is beyond the recommended full-scale range of 80% of most delta-sigma modulators. This value also does not represent the resolution of the signal. The signal resolution is determined by the modulator, and increasing the filter bit width will not offer any improved noise performance beyond the modulator capabilities.

NOTE: Because of the inherent architecture of the Sinc filter (Sinc1, Sinc2, SincFast, and Sinc3), the first few samples, depending upon the filter type, will be incorrect. Table 12-5 tabulates the number of incorrect samples after the filter is enabled and configured. This is an expected behavior.

Table 12-5. 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

SDFM Comparator interrupts (IELx and IEHx) 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.

In case of the SDFM data filter, each time the filter is enabled or reconfigured, or the filter is reset by the PWM sync pulse, or the filter is reset by SDDFPARMx.FEN, depending upon the filter type, there will be some incorrect samples as mentioned in Table 12-5.

12.5.1 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 SDIPARMx.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 SDIPARMx.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 SDIPARMx 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

The data filter with a 25-bit signed output value can be in the range of $-16,777,216$ to $16,777,215$. But, 16-bit signed output can support values only from $-32,768$ to $32,767$. Therefore, it is required to configure shift control bits (SDIPARMx.SH) to 9 to represent the data filter output correctly in 16-bit format. [Table 12-6](#) shows the configuration settings of shift control bits for different OSR and filter types.

Table 12-6. 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	9

WARNING

Configuring shift control bits incorrectly will result in getting wrong 16-bit data filter output.

12.5.2 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 μs

12.6 Interrupt Unit

Figure 12-9 shows the structure of the interrupt unit. Each SDFM module can generate a CPU interrupt. An SDFM interrupt can be triggered by any of these 16 interrupt sources:

- Four comparator low (IELx)

This interrupt source can be enabled when the master interrupt (SDCTL.MIE = 1) and the appropriate individual filter interrupts are enabled (SDCPARMx.IEL = 1). The Comparator compares the comparator filter output results continuously with the LLT threshold value. If the comparator filter output \leq LLT, the SDIFLG.IELx bit is set, triggering an under-value condition. This SDIFLG.IELx flag can be reset if the corresponding bit in the SDIFLGCLR register is set and the interrupt source is no longer active.

For example, since the comparator output changes only after every OSR SD-Cx clock cycle, writing to the SDIFLGCLR.IELx bit will clear the flag for only one SYSCLK cycle. On the next cycle, since the interrupt source is still active, the Comparator compares the comparator filter output with the lower threshold and sets the SDIFLG.IELx bit again. To avoid this situation, the comparator lower threshold setting should be changed to LLT = 0x0 before writing to the SDIFLGCLR.IELx bit.

- Four comparator high (IEHx)

This interrupt source can be enabled when the master interrupt (SDCTL.MIE = 1) and the appropriate individual filter interrupt are enabled (SDCPARMx.IEH = 1). The Comparator compares the comparator filter output results continuously with the HLT threshold value. If the comparator filter output \geq HLT, the SDIFLG.IEHx bit is set, triggering an over-value condition. This SDIFLG.IEHx flag can be reset if the corresponding bit in SDIFLGCLR register is set and the interrupt source is no longer active.

For example, since the comparator output changes only after every OSR SD-Cx clock cycle, writing to the SDIFLGCLR.IEHx bit will clear the flag for only one SYSCLK cycle. On the next cycle, since the interrupt source is still active, the Comparator compares the comparator filter output with the higher threshold and sets the SDIFLG.IEHx bit again. To avoid this situation, the comparator higher threshold setting should be changed to HLT = 0x7FFF before writing to the SDIFLGCLR.IEHx bit.

- Four modulator failure (MFx)

This interrupt source can be enabled when the master interrupt (SDCTL.MIE = 1) and the appropriate individual filter interrupt are enabled (SDCPARMx.MFIE = 1). When the modulator clock (SD-Cx) fails or goes missing, the appropriate MFx flag bit is set in the Interrupt Flag Register (SDIFLG). This flag will be reset if the Interrupt Register is cleared (by setting the corresponding bit in the SDIFLGCLR register) and the interrupt source is no longer active. If the modulator clock is failing (when the modulator clock is slower than 1/128th of the system clock SYSCLK), the appropriate MFx flag bit is set if the appropriate modulator flag interrupt enable bit (MFIE_x) and the master interrupt enable (MIE) is set.

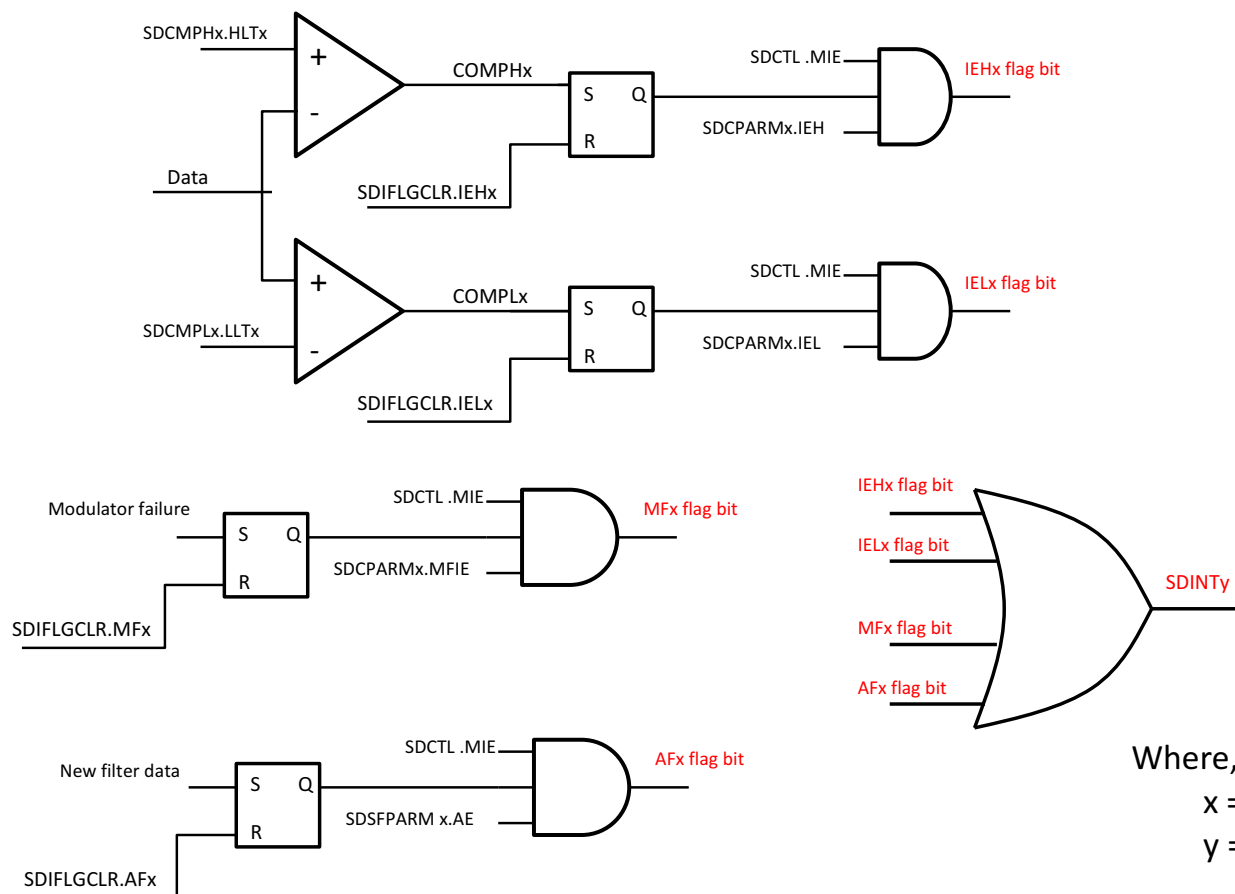
- Four filter data acknowledge (AFx)

This interrupt source can be enabled when the master interrupt (SDCTL.MIE = 1) and the appropriate individual filter interrupts are enabled (SDDFPARMx.AE = 1). When the data filter generates new data, the appropriate AFx flag bit is set in the Interrupt Flag Register (SDIFLG). This flag will be reset if the Interrupt Register is cleared (by setting the corresponding bit in the SDIFLGCLR register) and the interrupt source is no longer active. The acknowledge flags cannot be set if the data filter is disabled. Each acknowledge flag can be disabled if the Acknowledge Enable control bit (AE) in the appropriate Data Filter Parameter Register SDDFPARMx is set to low.

For some reason, if the user wishes to disable the data acknowledge event (AFx) and use a timer to generate an interrupt at the data filter rate to read filter data results, there is a possibility of reading a metastable filter data. In such cases, there are two possible options:

- Option 1: Make sure timer interrupts at a rate shown below
 - Timer interrupt time interval = Latency of data filter + 5 SD clock cycles
- Option 2: If the user disables PIE interrupts from SDFM, or if the Master Interrupt Enable (MIE) bit is disabled, a timer can be used to generate an interrupt at the data filter rate and filter data should be read after the data acknowledge event (AFx) is set

Figure 12-9. SDFM Interrupt Unit



12.7 Register Descriptions

The register descriptions are shown in the following table and subsections.

Table 12-7. General Registers

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDIFLG	0x5E00	0x5E80	2	Interrupt Flag Register	
SDIFLGCLR	0x5E02	0x5E82	2	Interrupt Flag Clear Register	
SDCTL	0x5E04	0x5E84	1	SD Control Register	YES
SDMFILEN	0x5E06	0x5E86	1	SD Master Filter Enable	YES
SDSTATUS	0x5E07	0x5E87	1	SD Status Register	YES

Table 12-8. Filter 1 Registers

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDCTLPARAM1	0x5E10	0x5E90	1	Control Parameter Register for Ch1	YES
SDDFPARAM1	0x5E11	0x5E91	1	Data Filter Parameter Register for Ch1	YES
SDIPARM1	0x5E12	0x5E92	1	Integer Parameter Register for Ch1	YES
SDCMPL1	0x5E13	0x5E93	1	High-level Threshold Register for Ch1	YES
SDCMPL1	0x5E14	0x5E94	1	Low-level Threshold Register for Ch1	YES
SDCPARM1	0x5E15	0x5E95	1	Comparator Parameter Register for Ch1	YES
SDDATA1	0x5E16	0x5E96	2	Filter Data Register (16- or 32-bit) for Ch1	

Table 12-9. Filter 2 Registers

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDCTLPARAM2	0x5E20	0x5EA0	1	Control Parameter Register for Ch2	YES
SDDFPARAM2	0x5E21	0x5EA1	1	Data Filter Parameter Register for Ch2	YES
SDIPARM2	0x5E22	0x5EA2	1	Integer Parameter Register for Ch2	YES
SDCMPL2	0x5E23	0x5EA3	1	High-level Threshold Register for Ch2	YES
SDCMPL2	0x5E24	0x5EA4	1	Low-level Threshold Register for Ch2	YES
SDCPARM2	0x5E25	0x5EA5	1	Comparator Parameter Register for Ch2	YES
SDDATA2	0x5E26	0x5EA6	2	Filter Data Register (16- or 32-bit) for Ch2	

Table 12-10. Filter 3 Registers

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDCTLPARAM3	0x5E30	0x5EB0	1	Control Parameter Register for Ch3	YES
SDDFPARAM3	0x5E31	0x5EB1	1	Data Filter Parameter Register for Ch3	YES
SDIPARM3	0x5E32	0x5EB2	1	Integer Parameter Register for Ch3	YES

Table 12-10. Filter 3 Registers (continued)

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDCMPH3	0x5E33	0x5EB3	1	High-level Threshold Register for Ch3	YES
SDCMPL3	0x5E34	0x5EB4	1	Low-level Threshold Register for Ch3	YES
SDCPARM3	0x5E35	0x5EB5	1	Comparator Parameter Register for Ch3	YES
SDDATA3	0x5E36	0x5EB6	2	Filter Data Register (16- or 32-bit) for Ch3	

Table 12-11. Filter 4 Registers

Name	SDFM1 address	SDFM2 address	Size (x16)	Description	EALLOW?
SDCTLARM4	0x5E40	0x5EC0	1	Control Parameter Register for Ch4	YES
SDDFPARM4	0x5E41	0x5EC1	1	Data Filter Parameter Register for Ch4	YES
SDIPARM4	0x5E42	0x5EC2	1	Integer Parameter Register for Ch4	YES
SDCMPH4	0x5E43	0x5EC3	1	High-level Threshold Register for Ch4	YES
SDCMPL4	0x5E44	0x5EC4	1	Low-level Threshold Register for Ch4	YES
SDCPARM4	0x5E45	0x5EC5	1	Comparator Parameter Register for Ch4	YES
SDDATA4	0x5E46	0x5EC6	2	Filter Data Register (16- or 32-bit) for Ch4	

12.8 Registers

12.8.1 SDFM Base Addresses

Table 12-12. SDFM Base Address Table

Device Registers	Register Name	Start Address	End Address
Sdfm1Regs	SDFM_REGS	0x0000_5E00	0x0000_5E7F
Sdfm2Regs	SDFM_REGS	0x0000_5E80	0x0000_5EFF

12.8.2 SDFM_REGS Registers

Table 12-13 lists the memory-mapped registers for the SDFM_REGS. All register offset addresses not listed in Table 12-13 should be considered as reserved locations and the register contents should not be modified.

Table 12-13. SDFM_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SDIFLG	Interrupt Flag Register		Go
2h	SDIFLGCLR	Interrupt Flag Clear Register		Go
4h	SDCTL	SD Control Register	EALLOW	Go
6h	SDFILEN	SD Master Filter Enable	EALLOW	Go
7h	SDSTATUS	SD Status Register	EALLOW	Go
10h	SDCTLPARM1	Control Parameter Register for Ch1	EALLOW	Go
11h	SDDFPARM1	Data Filter Parameter Register for Ch1	EALLOW	Go
12h	SDIPARM1	Integer Parameter Register for Ch1	EALLOW	Go
13h	SDCMPH1	High-level Threshold Register for Ch1	EALLOW	Go
14h	SDCMPL1	Low-level Threshold Register for Ch1	EALLOW	Go
15h	SDCPARM1	Comparator Parameter Register for Ch1	EALLOW	Go
16h	SDDATA1	Filter Data Register (16 or 32bit) for Ch1		Go
20h	SDCTLPARM2	Control Parameter Register for Ch2	EALLOW	Go
21h	SDDFPARM2	Data Filter Parameter Register for Ch2	EALLOW	Go
22h	SDIPARM2	Integer Parameter Register for Ch2	EALLOW	Go
23h	SDCMPH2	High-level Threshold Register for Ch2	EALLOW	Go
24h	SDCMPL2	Low-level Threshold Register for Ch2	EALLOW	Go
25h	SDCPARM2	Comparator Parameter Register for Ch2	EALLOW	Go
26h	SDDATA2	Filter Data Register (16 or 32bit) for Ch2		Go
30h	SDCTLPARM3	Control Parameter Register for Ch3	EALLOW	Go
31h	SDDFPARM3	Data Filter Parameter Register for Ch3	EALLOW	Go
32h	SDIPARM3	Integer Parameter Register for Ch3	EALLOW	Go
33h	SDCMPH3	High-level Threshold Register for Ch3	EALLOW	Go
34h	SDCMPL3	Low-level Threshold Register for Ch3	EALLOW	Go
35h	SDCPARM3	Comparator Parameter Register for Ch3	EALLOW	Go
36h	SDDATA3	Filter Data Register (16 or 32bit) for Ch3		Go
40h	SDCTLPARM4	Control Parameter Register for Ch4	EALLOW	Go
41h	SDDFPARM4	Data Filter Parameter Register for Ch4	EALLOW	Go
42h	SDIPARM4	Integer Parameter Register for Ch4	EALLOW	Go
43h	SDCMPH4	High-level Threshold Register for Ch4	EALLOW	Go
44h	SDCMPL4	Low-level Threshold Register for Ch4	EALLOW	Go
45h	SDCPARM4	Comparator Parameter Register for Ch4	EALLOW	Go
46h	SDDATA4	Filter Data Register (16 or 32bit) for Ch4		Go

12.8.2.1 SDIFLG Register (Offset = 0h) [reset = 0h]

SDIFLG is shown in [Figure 12-10](#) and described in [Table 12-14](#).

Interrupt Flag Register

Figure 12-10. SDIFLG Register

31	30	29	28	27	26	25	24
MIF	RESERVED						
R-0h	R-0h						
23	22	21	20	19	18	17	16
RESERVED							
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-14. SDIFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MIF	R	0h	Set whenever any interrupt (ACK1-4, MF1-4, IFL1-4, IFH1-4) is active
30-16	RESERVED	R	0h	Reserved
15	AF4	R	0h	0: No new data available from Filter 4 1: New data available from Filter 4
14	AF3	R	0h	0: No new data available from Filter 3 1: New data available from Filter 3
13	AF2	R	0h	0: No new data available from Filter 2 1: New data available from Filter 2
12	AF1	R	0h	0: No new data available from Filter 1 1: New data available from Filter 1
11	MF4	R	0h	0: Modulator is operating normally for Filter 4 1: Modulator failure for Filter 4
10	MF3	R	0h	0: Modulator is operating normally for Filter 3 1: Modulator failure for Filter 3
9	MF2	R	0h	0: Modulator is operating normally for Filter 2 1: Modulator failure for Filter 2
8	MF1	R	0h	0: Modulator is operating normally for Filter 1 1: Modulator failure for Filter 1
7	IFL4	R	0h	0: Comparator Filter 4 output is above the low limit threshold 1: Comparator Filter 4 output is equal to or below the low level threshold, if enabled
6	IFH4	R	0h	0: Comparator Filter 4 output is below the high limit threshold 1: Comparator Filter 4 output is equal to or above the high level threshold, if enabled
5	IFL3	R	0h	0: Comparator Filter 3 output is above the low limit threshold 1: Comparator Filter 3 output is equal to or below the low level threshold, if enabled

Table 12-14. SDIFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	IFH3	R	0h	0: Comparator Filter 3 output is below the high limit threshold 1: Comparator Filter 3 output is equal to or above the high level threshold, if enabled
3	IFL2	R	0h	0: Comparator Filter 2 output is above the low limit threshold 1: Comparator Filter 2 output is equal to or below the low level threshold, if enabled
2	IFH2	R	0h	0: Comparator Filter 2 output is below the high limit threshold 1: Comparator Filter 2 output is equal to or above the high level threshold, if enabled
1	IFL1	R	0h	0: Comparator Filter 1 output is above the low limit threshold 1: Comparator Filter 1 output is equal to or below the low level threshold, if enabled
0	IFH1	R	0h	0: Comparator Filter 1 output is below the high limit threshold 1: Comparator Filter 1 output is equal to or above the high level threshold, if enabled

12.8.2.2 SDIFLGCLR Register (Offset = 2h) [reset = 0h]

SDIFLGCLR is shown in [Figure 12-11](#) and described in [Table 12-15](#).

Interrupt Flag Clear Register

Figure 12-11. SDIFLGCLR Register

31	30	29	28	27	26	25	24
MIF	RESERVED						
R=0/W=1-0h	R-0h						
23	22	21	20	19	18	17	16
RESERVED							
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-15. SDIFLGCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MIF	R=0/W=1	0h	Flag-clear bit for SDFM Master Interrupt flag. Write 1 to clear MIF. 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)
30-16	RESERVED	R	0h	Reserved
15	AF4	R=0/W=1	0h	SD Module Interrupt Flag Clear Bits: Writing a "1" will clear the respective flag bit in the SDINTFLG 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. Flag-clear bit for Acknowledge flag for Filter 4
14	AF3	R=0/W=1	0h	Flag-clear bit for Acknowledge flag for Filter 3
13	AF2	R=0/W=1	0h	Flag-clear bit for Acknowledge flag for Filter 2
12	AF1	R=0/W=1	0h	Flag-clear bit for Acknowledge flag for Filter 1
11	MF4	R=0/W=1	0h	Flag-clear bit for Modulator Failure for Filter 4
10	MF3	R=0/W=1	0h	Flag-clear bit for Modulator Failure for Filter 3
9	MF2	R=0/W=1	0h	Flag-clear bit for Modulator Failure for Filter 2
8	MF1	R=0/W=1	0h	Flag-clear bit for Modulator Failure for Filter 1
7	IFL4	R=0/W=1	0h	Flag-clear bit for Low-Level Interrupt flag Filter 4
6	IFH4	R=0/W=1	0h	Flag-clear bit for High-level Interrupt flag Filter 4
5	IFL3	R=0/W=1	0h	Flag-clear bit for Low-Level Interrupt flag Filter 3
4	IFH3	R=0/W=1	0h	Flag-clear bit for High-level Interrupt flag Filter 3
3	IFL2	R=0/W=1	0h	Flag-clear bit for Low-Level Interrupt flag Filter 2
2	IFH2	R=0/W=1	0h	Flag-clear bit for High-level Interrupt flag Filter 2
1	IFL1	R=0/W=1	0h	Flag-clear bit for Low-Level Interrupt flag Filter 1
0	IFH1	R=0/W=1	0h	Flag-clear bit for High-level Interrupt flag Filter 1

12.8.2.3 SDCTL Register (Offset = 4h) [reset = 0h]

SDCTL is shown in [Figure 12-12](#) and described in [Table 12-16](#).

SD Control Register

Figure 12-12. SDCTL Register

15	14	13	12	11	10	9	8
RESERVED	RESERVED	MIE	RESERVED				
R-0h	R-0h	R/W-0h	R-0h				
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-16. SDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	MIE	R/W	0h	Master interrupt enable. 0: Interrupt pin and interrupt flags are blocked (interrupt pin INT always inactive). 1: Interrupt pin and interrupt flags are not blocked and can be set and reset (if individually enabled).
12-0	RESERVED	R	0h	Reserved

12.8.2.4 SDMFILEN Register (Offset = 6h) [reset = 0h]

SDMFILEN is shown in [Figure 12-13](#) and described in [Table 12-17](#).

SD Master Filter Enable

Figure 12-13. SDMFILEN Register

15	14	13	12	11	10	9	8
RESERVED			RESERVED	MFE	RESERVED	RESERVED	RESERVED
R-0h			R-0h	R/W-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED			RESERVED			
R-0h	R-0h			R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-17. SDMFILEN Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	MFE	R/W	0h	Master Filter Enable. Functionally AND'ed with bit FEN in the Data Filter Parameter Register 0: Data filter units of all filter modules are disabled. 1: Data filter units can be enabled if bit FEN is '1'.
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8-7	RESERVED	R	0h	Reserved
6-4	RESERVED	R	0h	Reserved
3-0	RESERVED	R	0h	Reserved

12.8.2.5 SDSTATUS Register (Offset = 7h) [reset = 0h]

SDSTATUS is shown in [Figure 12-14](#) and described in [Table 12-18](#).

SD Status Register

Figure 12-14. SDSTATUS Register

15	14	13	12	11	10	9	8
MAL4	MAL3	MAL2	MAL1	MAF4	MAF3	MAF2	MAF1
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							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-18. SDSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
15	MAL4	R	0h	Manchester locked status for filter module 4. 0: The automatic Manchester decoder calibration is working properly 1: The automatic Manchester decoder calibration has not been able to perform a successful calibration
14	MAL3	R	0h	Manchester locked status for filter module 3. 0: The automatic Manchester decoder calibration is working properly 1: The automatic Manchester decoder calibration has not been able to perform a successful calibration
13	MAL2	R	0h	Manchester locked status for filter module 2. 0: The automatic Manchester decoder calibration is working properly 1: The automatic Manchester decoder calibration has not been able to perform a successful calibration
12	MAL1	R	0h	Manchester locked status for filter module 1. 0: The automatic Manchester decoder calibration is working properly 1: The automatic Manchester decoder calibration has not been able to perform a successful calibration
11	MAF4	R	0h	Manchester failure status for filter module 4. 0: The automatic Manchester decoder calibration has worked properly since last read access 1: The automatic Manchester decoder has detected problems since last read access
10	MAF3	R	0h	Manchester failure status for filter module 3. 0: The automatic Manchester decoder calibration has worked properly since last read access 1: The automatic Manchester decoder has detected problems since last read access
9	MAF2	R	0h	Manchester failure status for filter module 2. 0: The automatic Manchester decoder calibration has worked properly since last read access 1: The automatic Manchester decoder has detected problems since last read access
8	MAF1	R	0h	Manchester failure status for filter module 1. 0: The automatic Manchester decoder calibration has worked properly since last read access 1: The automatic Manchester decoder has detected problems since last read access
7-0	RESERVED	R	0h	Reserved

12.8.2.6 SDCTLPARM1 Register (Offset = 10h) [reset = 0h]

SDCTLPARM1 is shown in [Figure 12-15](#) and described in [Table 12-19](#).

Control Parameter Register for Ch1

Figure 12-15. SDCTLPARM1 Register

15	14	13	12	11	10	9	8
MS							
R-0h							
7	6	5	4	3	2	1	0
MS			RESERVED	RESERVED	RESERVED	MOD	
R-0h			R-0h	R-0h	R-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-19. SDCTLPARM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	MS	R	0h	Manchester Status MS10:MS0
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1-0	MOD	R/W	0h	Delta-Sigma Modulator mode 00: The clock speed is equal to the data rate from the modulator 01: The clock rate is half of the data rate from the modulator 10: The data from the modulator is Manchester decoded 11: The clock rate is twice the data rate of the modulator

12.8.2.7 SDDFPARM1 Register (Offset = 11h) [reset = 0h]

SDDFPARM1 is shown in [Figure 12-16](#) and described in [Table 12-20](#).

Data Filter Parameter Register for Ch1

Figure 12-16. SDDFPARM1 Register

15	14	13	12	11	10	9	8
RESERVED			FILRESEN	SST		AE	FEN
R-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-20. SDDFPARM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	FILRESEN	R/W	0h	Data Filter Reset enable for External Reset typ from PWM Compare output. 0: Data filter cannot be reset by external PWM compare output 1: Data filter can be reset by external PWM compare output
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
9	AE	R/W	0h	Acknowledge enable. 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter
8	FEN	R/W	0h	Filter enable. 0: The filter is disabled and no data is produced 1: The filter is enabled and data are produced in the Data filter
7-0	DOSR	R/W	0h	Oversampling ratio. The actual rate is DOSR + 1. These bits set the oversampling ratio of the filter. 0x0FF represents an oversampling ratio of 256.

12.8.2.8 SDIPARM1 Register (Offset = 12h) [reset = 0h]

SDIPARM1 is shown in [Figure 12-17](#) and described in [Table 12-21](#).

Integer Parameter Register for Ch1

Figure 12-17. SDIPARM1 Register

15	14	13	12	11	10	9	8
SH					DR	RESERVED	RESERVED
R/W-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED						
R-0h	R/W-0h						

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-21. SDIPARM1 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.
10	DR	R/W	0h	Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R	0h	Reserved
6-0	RESERVED	R/W	0h	Reserved

12.8.2.9 SDCMPH1 Register (Offset = 13h) [reset = 0h]

SDCMPH1 is shown in [Figure 12-18](#) and described in [Table 12-22](#).

High-level Threshold Register for Ch1

Figure 12-18. SDCMPH1 Register

15	14	13	12	11	10	9	8
RESERVED	HLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
HLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-22. SDCMPH1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	HLT	R/W	0h	Unsigned high-level threshold for the comparator filter output.

12.8.2.10 SDCMPL1 Register (Offset = 14h) [reset = 0h]

SDCMPL1 is shown in [Figure 12-19](#) and described in [Table 12-23](#).

Low-level Threshold Register for Ch1

Figure 12-19. SDCMPL1 Register

15	14	13	12	11	10	9	8
RESERVED	LLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-23. SDCMPL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output.

12.8.2.11 SDCPARM1 Register (Offset = 15h) [reset = 0h]

SDCPARM1 is shown in [Figure 12-20](#) and described in [Table 12-24](#).

Comparator Parameter Register for Ch1

Figure 12-20. SDCPARM1 Register

15	14	13	12	11	10	9	8
RESERVED						MFIE	CS1_CS0
R-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				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-24. SDCPARM1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	MFIE	R/W	0h	0: The modulator failure flag as well as the output INT is disabled for this particular flag 1: The modulator failure flag is enabled
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
6	IEL	R/W	0h	Low-level interrupt enable. 0: The low-level interrupt flag as well as the output INT is disabled for this particular flag 1: The low-level interrupt flag is enabled
5	IEH	R/W	0h	High-level interrupt enable. 0: The high-level interrupt flag as well as the output INT is disabled for this particular flag 1: The high-level interrupt flag is enabled
4-0	COSR	R/W	0h	Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 31.

12.8.2.12 SDDATA1 Register (Offset = 16h) [reset = 0h]

SDDATA1 is shown in [Figure 12-21](#) and described in [Table 12-25](#).

Filter Data Register (16 or 32bit) for Ch1

Figure 12-21. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-25. SDDATA1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode
15-0	DATA16	R	0h	16-bit Data in 16b mode, Lo-order 16b in 32b mode

12.8.2.13 SDCTLPARM2 Register (Offset = 20h) [reset = 0h]

SDCTLPARM2 is shown in [Figure 12-22](#) and described in [Table 12-26](#).

Control Parameter Register for Ch2

Figure 12-22. SDCTLPARM2 Register

15	14	13	12	11	10	9	8
MS							
R-0h							
7	6	5	4	3	2	1	0
MS			RESERVED	RESERVED	RESERVED	MOD	
R-0h			R-0h	R-0h	R-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-26. SDCTLPARM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	MS	R	0h	Manchester Status MS10:MS0
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1-0	MOD	R/W	0h	Delta-Sigma Modulator mode 00: The clock speed is equal to the data rate from the modulator 01: The clock rate is half of the data rate from the modulator 10: The data from the modulator is Manchester decoded 11: The clock rate is twice the data rate of the modulator

12.8.2.14 SDDFPARM2 Register (Offset = 21h) [reset = 0h]

SDDFPARM2 is shown in [Figure 12-23](#) and described in [Table 12-27](#).

Data Filter Parameter Register for Ch2

Figure 12-23. SDDFPARM2 Register

15	14	13	12	11	10	9	8
RESERVED			FILRESEN	SST		AE	FEN
R-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-27. SDDFPARM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	FILRESEN	R/W	0h	Data Filter Reset enable for External Reset typ from PWM Compare output. 0: Data filter cannot be reset by external PWM compare output 1: Data filter can be reset by external PWM compare output
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
9	AE	R/W	0h	Acknowledge enable. 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter
8	FEN	R/W	0h	Filter enable. 0: The filter is disabled and no data is produced 1: The filter is enabled and data are produced in the Data filter
7-0	DOSR	R/W	0h	Oversampling ratio. The actual rate is DOSR + 1. These bits set the oversampling ratio of the filter. 0x0FF represents an oversampling ratio of 256.

12.8.2.15 SDIPARM2 Register (Offset = 22h) [reset = 0h]

SDIPARM2 is shown in [Figure 12-24](#) and described in [Table 12-28](#).

Integer Parameter Register for Ch2

Figure 12-24. SDIPARM2 Register

15	14	13	12	11	10	9	8
SH					DR	RESERVED	RESERVED
R/W-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED						
R-0h	R/W-0h						

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-28. SDIPARM2 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.
10	DR	R/W	0h	Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R	0h	Reserved
6-0	RESERVED	R/W	0h	Reserved

12.8.2.16 SDCMPH2 Register (Offset = 23h) [reset = 0h]

SDCMPH2 is shown in [Figure 12-25](#) and described in [Table 12-29](#).

High-level Threshold Register for Ch2

Figure 12-25. SDCMPH2 Register

15	14	13	12	11	10	9	8
RESERVED	HLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
HLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-29. SDCMPH2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	HLT	R/W	0h	Unsigned high-level threshold for the comparator filter output.

12.8.2.17 SDCMPL2 Register (Offset = 24h) [reset = 0h]

SDCMPL2 is shown in [Figure 12-26](#) and described in [Table 12-30](#).

Low-level Threshold Register for Ch2

Figure 12-26. SDCMPL2 Register

15	14	13	12	11	10	9	8
RESERVED	LLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-30. SDCMPL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output.

12.8.2.18 SDCPARM2 Register (Offset = 25h) [reset = 0h]

SDCPARM2 is shown in [Figure 12-27](#) and described in [Table 12-31](#).

Comparator Parameter Register for Ch2

Figure 12-27. SDCPARM2 Register

15	14	13	12	11	10	9	8
RESERVED						MFIE	CS1_CS0
R-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				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-31. SDCPARM2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	MFIE	R/W	0h	0: The modulator failure flag as well as the output INT is disabled for this particular flag 1: The modulator failure flag is enabled
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
6	IEL	R/W	0h	Low-level interrupt enable. 0: The low-level interrupt flag as well as the output INT is disabled for this particular flag 1: The low-level interrupt flag is enabled
5	IEH	R/W	0h	High-level interrupt enable. 0: The high-level interrupt flag as well as the output INT is disabled for this particular flag 1: The high-level interrupt flag is enabled
4-0	COSR	R/W	0h	Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 31.

12.8.2.19 SDDATA2 Register (Offset = 26h) [reset = 0h]

SDDATA2 is shown in [Figure 12-28](#) and described in [Table 12-32](#).

Filter Data Register (16 or 32bit) for Ch2

Figure 12-28. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-32. SDDATA2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode
15-0	DATA16	R	0h	16-bit Data in 16b mode, Lo-order 16b in 32b mode

12.8.2.20 SDCTLPARM3 Register (Offset = 30h) [reset = 0h]

SDCTLPARM3 is shown in [Figure 12-29](#) and described in [Table 12-33](#).

Control Parameter Register for Ch3

Figure 12-29. SDCTLPARM3 Register

15	14	13	12	11	10	9	8
MS							
R-0h							
7	6	5	4	3	2	1	0
MS			RESERVED	RESERVED	RESERVED	MOD	
R-0h			R-0h	R-0h	R-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-33. SDCTLPARM3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	MS	R	0h	Manchester Status MS10:MS0
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1-0	MOD	R/W	0h	Delta-Sigma Modulator mode 00: The clock speed is equal to the data rate from the modulator 01: The clock rate is half of the data rate from the modulator 10: The data from the modulator is Manchester decoded 11: The clock rate is twice the data rate of the modulator

12.8.2.21 SDDFPARM3 Register (Offset = 31h) [reset = 0h]

SDDFPARM3 is shown in [Figure 12-30](#) and described in [Table 12-34](#).

Data Filter Parameter Register for Ch3

Figure 12-30. SDDFPARM3 Register

15	14	13	12	11	10	9	8
RESERVED			FILRESEN	SST		AE	FEN
R-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-34. SDDFPARM3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	FILRESEN	R/W	0h	Data Filter Reset enable for External Reset typ from PWM Compare output. 0: Data filter cannot be reset by external PWM compare output 1: Data filter can be reset by external PWM compare output
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
9	AE	R/W	0h	Acknowledge enable. 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter
8	FEN	R/W	0h	Filter enable. 0: The filter is disabled and no data is produced 1: The filter is enabled and data are produced in the data filter
7-0	DOSR	R/W	0h	Oversampling ratio. The actual rate is DOSR + 1. These bits set the oversampling ratio of the filter. 0x0FF represents an oversampling ratio of 256.

12.8.2.22 SDIPARM3 Register (Offset = 32h) [reset = 0h]

SDIPARM3 is shown in [Figure 12-31](#) and described in [Table 12-35](#).

Integer Parameter Register for Ch3

Figure 12-31. SDIPARM3 Register

15	14	13	12	11	10	9	8
SH					DR	RESERVED	RESERVED
R/W-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED						
R-0h	R/W-0h						

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-35. SDIPARM3 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.
10	DR	R/W	0h	Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R	0h	Reserved
6-0	RESERVED	R/W	0h	Reserved

12.8.2.23 SDCMPH3 Register (Offset = 33h) [reset = 0h]

SDCMPH3 is shown in [Figure 12-32](#) and described in [Table 12-36](#).

High-level Threshold Register for Ch3

Figure 12-32. SDCMPH3 Register

15	14	13	12	11	10	9	8
RESERVED	HLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
HLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-36. SDCMPH3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	HLT	R/W	0h	Unsigned high-level threshold for the comparator filter output.

12.8.2.24 SDCMPL3 Register (Offset = 34h) [reset = 0h]

SDCMPL3 is shown in [Figure 12-33](#) and described in [Table 12-37](#).

Low-level Threshold Register for Ch3

Figure 12-33. SDCMPL3 Register

15	14	13	12	11	10	9	8
RESERVED	LLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-37. SDCMPL3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output.

12.8.2.25 SDCPARM3 Register (Offset = 35h) [reset = 0h]

SDCPARM3 is shown in [Figure 12-34](#) and described in [Table 12-38](#).

Comparator Parameter Register for Ch3

Figure 12-34. SDCPARM3 Register

15	14	13	12	11	10	9	8
RESERVED						MFIE	CS1_CS0
R-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				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-38. SDCPARM3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	MFIE	R/W	0h	0: The modulator failure flag as well as the output INT is disabled for this particular flag 1: The modulator failure flag is enabled
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
6	IEL	R/W	0h	Low-level interrupt enable. 0: The low-level interrupt flag as well as the output INT is disabled for this particular flag 1: The low-level interrupt flag is enabled
5	IEH	R/W	0h	High-level interrupt enable. 0: The high-level interrupt flag as well as the output INT is disabled for this particular flag 1: The high-level interrupt flag is enabled
4-0	COSR	R/W	0h	Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 31.

12.8.2.26 SDDATA3 Register (Offset = 36h) [reset = 0h]

SDDATA3 is shown in [Figure 12-35](#) and described in [Table 12-39](#).

Filter Data Register (16 or 32bit) for Ch3

Figure 12-35. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-39. SDDATA3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode
15-0	DATA16	R	0h	16-bit Data in 16b mode, Lo-order 16b in 32b mode

12.8.2.27 SDCTLPARM4 Register (Offset = 40h) [reset = 0h]

SDCTLPARM4 is shown in [Figure 12-36](#) and described in [Table 12-40](#).

Control Parameter Register for Ch4

Figure 12-36. SDCTLPARM4 Register

15	14	13	12	11	10	9	8
MS							
R-0h							
7	6	5	4	3	2	1	0
MS			RESERVED	RESERVED	RESERVED	MOD	
R-0h			R-0h	R-0h	R-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-40. SDCTLPARM4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	MS	R	0h	Manchester Status MS10:MS0
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1-0	MOD	R/W	0h	Delta-Sigma Modulator mode 00: The clock speed is equal to the data rate from the modulator 01: The clock rate is half of the data rate from the modulator 10: The data from the modulator is Manchester decoded 11: The clock rate is twice the data rate of the modulator

12.8.2.28 SDDFPARM4 Register (Offset = 41h) [reset = 0h]

SDDFPARM4 is shown in [Figure 12-37](#) and described in [Table 12-41](#).

Data Filter Parameter Register for Ch4

Figure 12-37. SDDFPARM4 Register

15	14	13	12	11	10	9	8
RESERVED			FILRESEN	SST		AE	FEN
R-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOSR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-41. SDDFPARM4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	FILRESEN	R/W	0h	Data Filter Reset enable for External Reset typ from PWM Compare output. 0: Data filter cannot be reset by external PWM compare output 1: Data filter can be reset by external PWM compare output
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
9	AE	R/W	0h	Acknowledge enable. 0: Acknowledge flag is disabled for the particular filter 1: Acknowledge flag is enabled for the particular filter
8	FEN	R/W	0h	Filter enable. 0: The filter is disabled and no data is produced 1: The filter is enabled and data are produced in the sinc filter
7-0	DOSR	R/W	0h	Oversampling ratio. The actual rate is DOSR + 1. These bits set the oversampling ratio of the filter. 0x0FF represents an oversampling ratio of 256.

12.8.2.29 SDIPARM4 Register (Offset = 42h) [reset = 0h]

SDIPARM4 is shown in [Figure 12-38](#) and described in [Table 12-42](#).

Integer Parameter Register for Ch4

Figure 12-38. SDIPARM4 Register

15	14	13	12	11	10	9	8
SH					DR	RESERVED	RESERVED
R/W-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED						
R-0h	R/W-0h						

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-42. SDIPARM4 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.
10	DR	R/W	0h	Data representation 0: Data stored in 16b 2's complement 1: Data stored in 32b 2's complement
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R	0h	Reserved
6-0	RESERVED	R/W	0h	Reserved

12.8.2.30 SDCMPH4 Register (Offset = 43h) [reset = 0h]

SDCMPH4 is shown in [Figure 12-39](#) and described in [Table 12-43](#).

High-level Threshold Register for Ch4

Figure 12-39. SDCMPH4 Register

15	14	13	12	11	10	9	8
RESERVED	HLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
HLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-43. SDCMPH4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	HLT	R/W	0h	Unsigned high-level threshold for the comparator filter output.

12.8.2.31 SDCMPL4 Register (Offset = 44h) [reset = 0h]

SDCMPL4 is shown in [Figure 12-40](#) and described in [Table 12-44](#).

Low-level Threshold Register for Ch4

Figure 12-40. SDCMPL4 Register

15	14	13	12	11	10	9	8
RESERVED	LLT						
R-0h	R/W-0h						
7	6	5	4	3	2	1	0
LLT							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-44. SDCMPL4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14-0	LLT	R/W	0h	Unsigned low-level threshold for the comparator filter output.

12.8.2.32 SDCPARM4 Register (Offset = 45h) [reset = 0h]

SDCPARM4 is shown in [Figure 12-41](#) and described in [Table 12-45](#).

Comparator Parameter Register for Ch4

Figure 12-41. SDCPARM4 Register

15	14	13	12	11	10	9	8
RESERVED						MFIE	CS1_CS0
R-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				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-45. SDCPARM4 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	MFIE	R/W	0h	0: The modulator failure flag as well as the output INT is disabled for this particular flag 1: The modulator failure flag is enabled
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
6	IEL	R/W	0h	Low-level interrupt enable. 0: The low-level interrupt flag as well as the output INT is disabled for this particular flag 1: The low-level interrupt flag is enabled
5	IEH	R/W	0h	High-level interrupt enable. 0: The high-level interrupt flag as well as the output INT is disabled for this particular flag 1: The high-level interrupt flag is enabled
4-0	COSR	R/W	0h	Oversampling ratio. The actual rate is COSR + 1. These bits set the oversampling ratio of the filter. 0x1F represents an oversampling ratio of 31.

12.8.2.33 SDDATA4 Register (Offset = 46h) [reset = 0h]

SDDATA4 is shown in [Figure 12-42](#) and described in [Table 12-46](#).

Filter Data Register (16 or 32bit) for Ch4

Figure 12-42. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 12-46. SDDATA4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	DATA32HI	R	0h	Hi-order 16b in 32b mode
15-0	DATA16	R	0h	16-bit Data in 16b mode, Lo-order 16b in 32b mode

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. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* ([SPRU566](#)) 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.

Topic	Page
13.1 Introduction	1467
13.2 Configuring Device Pins	1472
13.3 ePWM Submodules	1473
13.4 Applications to Power Topologies	1539
13.5 Registers	1557

13.1 Introduction

This chapter includes an overview of the module and information about each of its submodules:

- Time-Base Module
- Counter Compare Module
- Action Qualifier Module
- Dead-Band Generator Module
- PWM Chopper (PC) Module
- Trip Zone Module
- Event Trigger Module
- Digital Compare Module

The ePWM Type 4 is functionally compatible to the Type 2 module (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.

- **Dead-Band Generator Module Enhancements**

Shadowing of DBCTL register to allow dynamic configuration changes

- **One Shot and Global Reload of Registers**

The ePWM Type 4 allows one shot and global reload capability from shadow to active registers to avoid partial reloads in, for example, multi-phase applications. It also allows programmable pre-scale of shadow to active reload events.

- **Trip Zone Module 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 module to support certain power converter switching techniques like valley switching.

- **Digital Compare Module 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 the Type 1 module. 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 Module 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 the *HRPWM* chapter in this manual.

- **Counter Compare Module Enhancements**

The ePWM Type 2 allows Interrupts and SOC events to be generated by additional counter compares CMPC and CMPD.

- **Event Trigger Module 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 Module 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 needs to 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 modules 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 module 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.

13.1.1 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 13-1](#). 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 the device-specific *High-Resolution Pulse Width Modulator (HRPWM)* chapter. 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 modules (eCAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

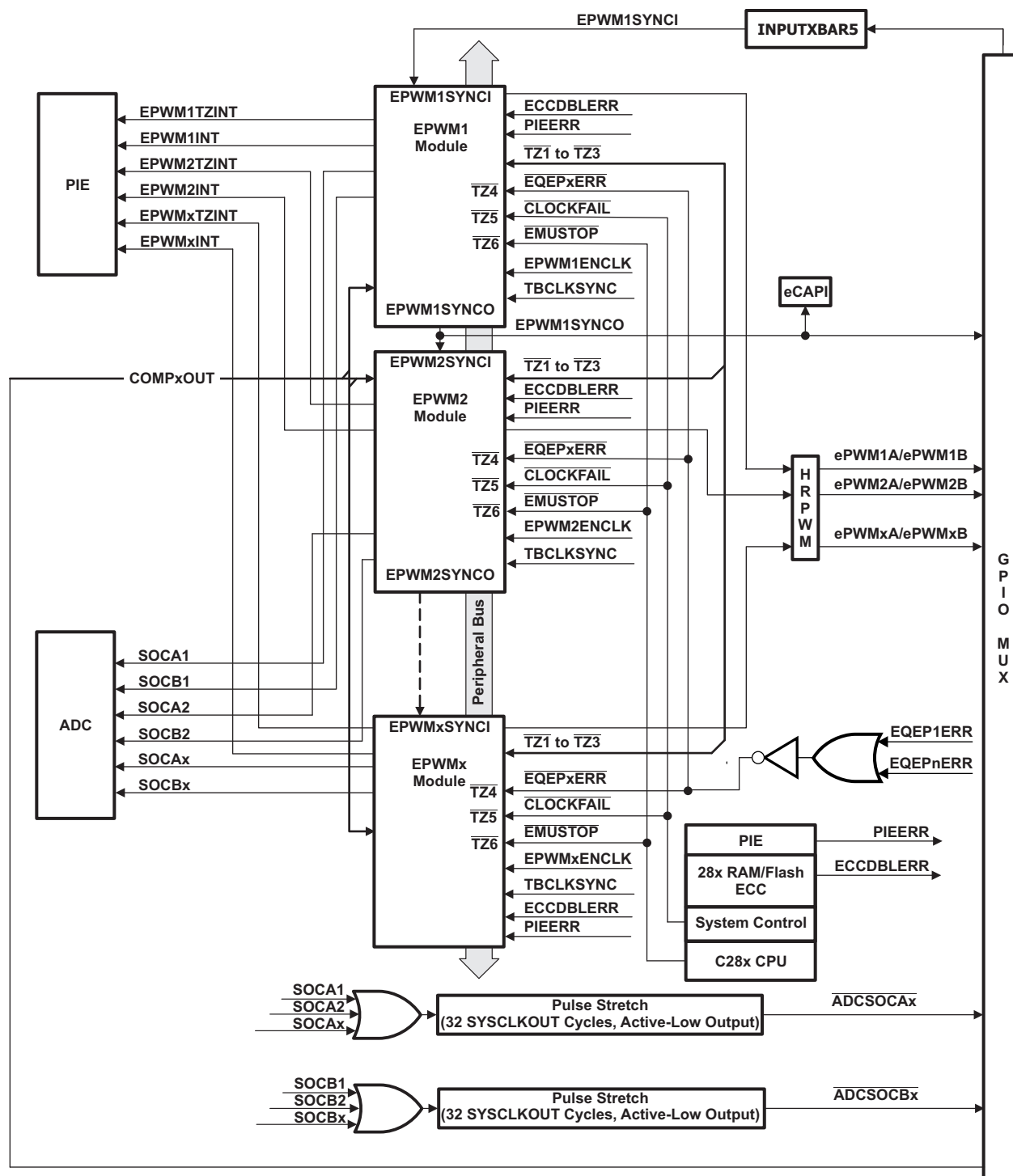
Each ePWM module supports the following features:

- 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 13-1. The signals are described in detail in subsequent sections.

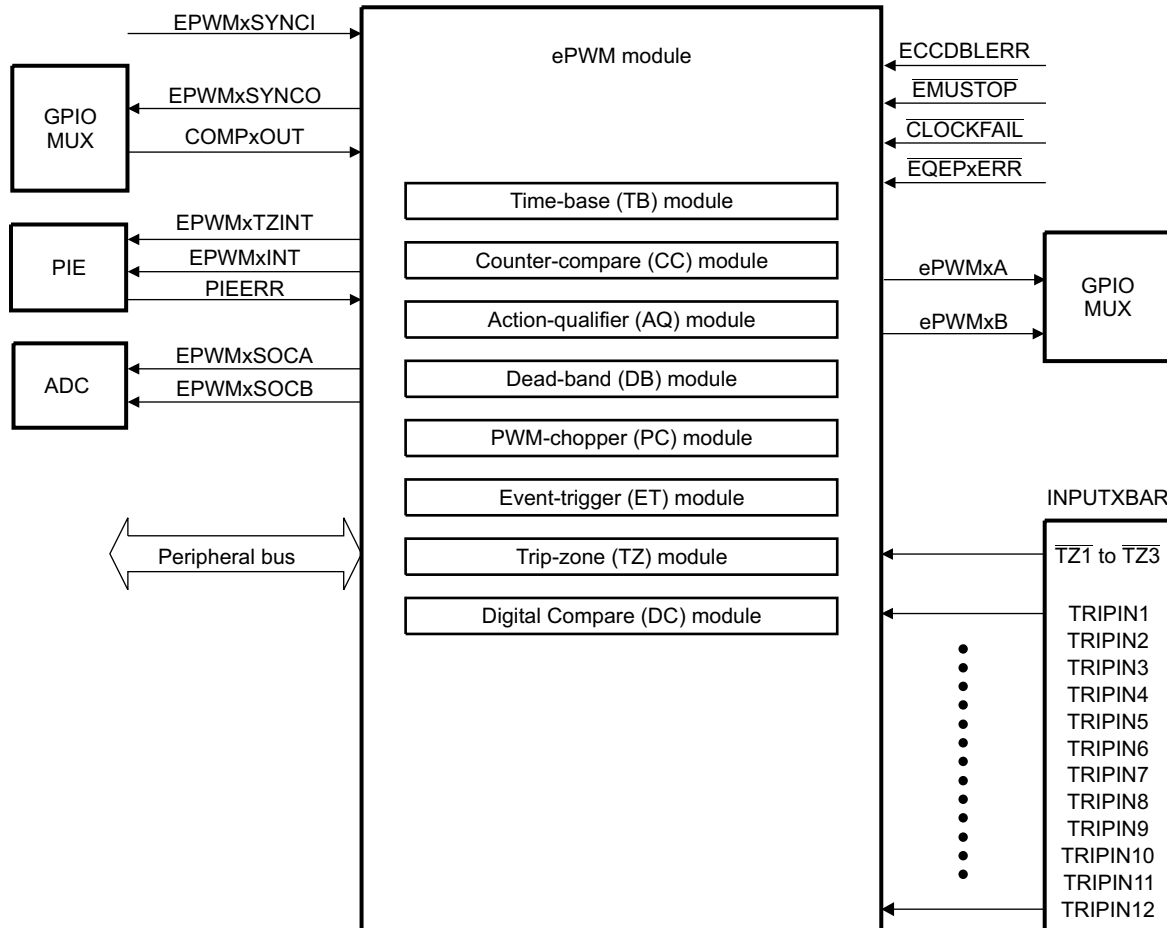
Figure 13-1. Multiple ePWM Modules



A This signal exists only on devices with an eQEP module.

The order in which the ePWM modules are connected may differ from what is shown in [Figure 13-1](#). See [Section 13.3.2.3.3](#) 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 13-2](#).

Figure 13-2. Submodules and Signal Connections for an ePWM Module



[Figure 13-3](#) 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 module 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 13-49](#). TZ4 is connected to an inverted EQEPx error signal (EQEPxERR), which can be generated from any one of the EQEP module (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) and output (EPWMxSYNCO) signals.**

The synchronization signals daisy chain the ePWM modules 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 modules 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 modules can also send their EPWMxSYNCOOUT signal to a GPIO. For more information, see [Section 13.3.2.3.3](#).

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

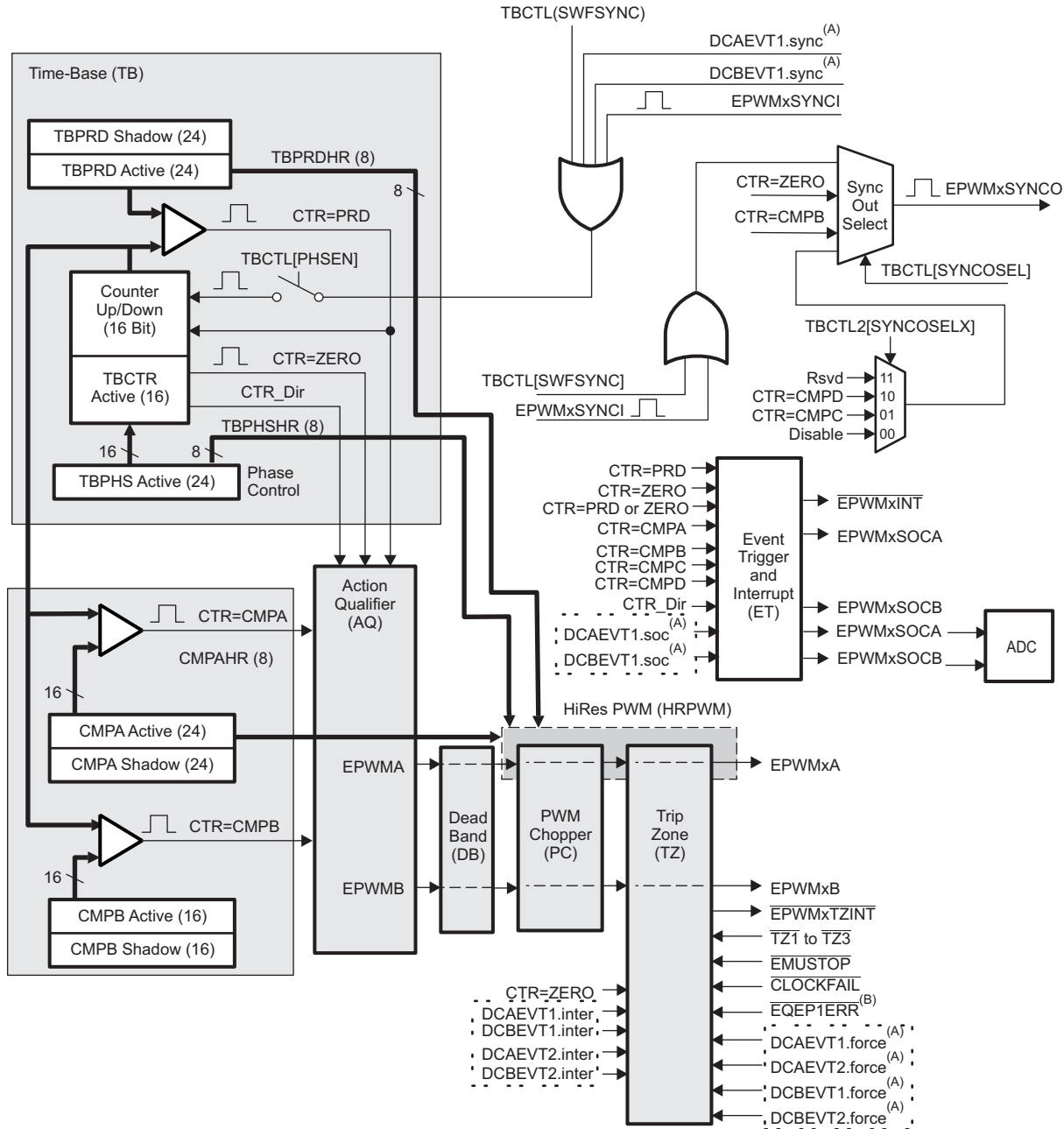
Each ePWM module has two ADC start of conversion signals. Any ePWM module can trigger a start of conversion. Whichever event triggers the start of conversion is configured in the event-trigger submodule of the ePWM.

- **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 13-3. ePWM Submodules and Critical Internal Signal Interconnects


A These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs.

B This signal exists only on devices with eQEP module

13.2 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 EXTSYNIN. 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.

13.3 ePWM Submodules

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

13.3.1 Overview

[Table 13-1](#) 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 13.3.3](#) for relevant details.

Table 13-1. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option
Time Base (TB)	<ul style="list-style-type: none"> Scale the time-base clock (TBCLK) relative to the EPWM clock (EPWMCLK). Configure the PWM time-base counter (TBCTR) frequency or period. Set the mode for the time-base counter: <ul style="list-style-type: none"> count-up mode: used for asymmetric PWM count-down mode: used for asymmetric PWM count-up-and-down mode: used for symmetric PWM Configure the time-base phase relative to another ePWM module. Synchronize the time-base counter between modules through hardware or software. Configure the direction (up or down) of the time-base counter after a synchronization event. Simultaneous writes to the TBPRD registers on all PWM's corresponding to the configuration on EPWMXLINK. Configure how the time-base counter will behave when the device is halted by an emulator. Specify the source for the synchronization output of the ePWM module: <ul style="list-style-type: none"> Synchronization input signal Time-base counter equal to zero Time-base counter equal to counter-compare B (CMPB) No output synchronization signal generated. Configure one shot and global reload of registers in this module.
Counter Compare (CC)	<ul style="list-style-type: none"> Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB Specify the time at which switching events occur on the EPWMxA or EPWMxB output Specify the programmable delay for interrupt and SOC generation with additional comparators Simultaneous writes to the CMPA, CMPB, CMPC, CMPD registers on all PWM's corresponding to the configuration on EPWMXLINK. Configure one shot and global reload of registers in this module.
Action Qualifier (AQ)	<ul style="list-style-type: none"> 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"> No action taken Output EPWMxA and/or EPWMxB switched high Output EPWMxA and/or EPWMxB switched low Output EPWMxA and/or EPWMxB toggled Force the PWM output state through software control Configure and control the PWM dead band through software Configure one shot and global reload of registers in this module.

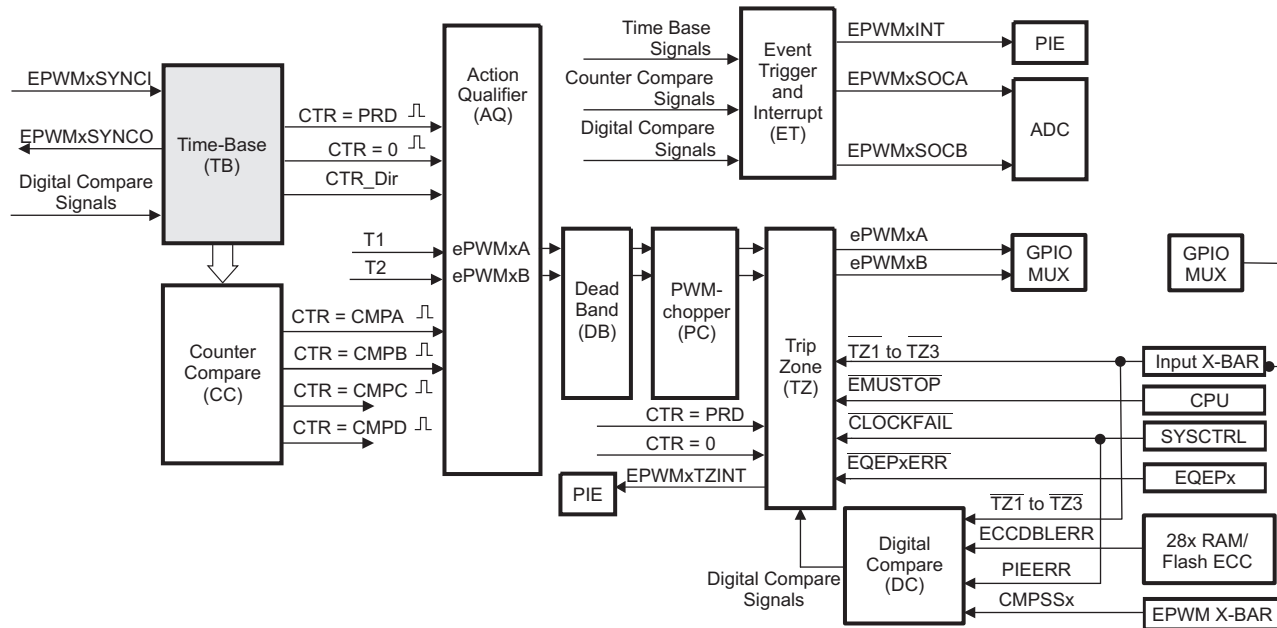
Table 13-1. Submodule Configuration Parameters (continued)

Submodule	Configuration Parameter or Option
Dead Band (DB)	<ul style="list-style-type: none"> Control of traditional complementary dead-band relationship between upper and lower switches Specify the output rising-edge-delay value Specify the output falling-edge delay value Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification. Option to enable half-cycle clocking for double resolution. Allow ePWMxB phase shifting with respect to the ePWMxA output. Configure one shot and global reload of registers in this module.
PWM Chopper (PC)	<ul style="list-style-type: none"> Create a chopping (carrier) frequency. Pulse width of the first pulse in the chopped pulse train. Duty cycle of the second and subsequent pulses. Bypass the PWM chopper module entirely. In this case the PWM waveform is passed through without modification.
Trip Zone (TZ)	<ul style="list-style-type: none"> Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events. Specify the trip action taken when a fault occurs: <ul style="list-style-type: none"> Force EPWMxA and/or EPWMxB high Force EPWMxA and/or EPWMxB low Force EPWMxA and/or EPWMxB to a high-impedance state Configure EPWMxA and/or EPWMxB to ignore any trip condition. Configure how often the ePWM will react to each trip-zone signal: <ul style="list-style-type: none"> One-shot Cycle-by-cycle Enable the trip-zone to initiate an interrupt. Bypass the trip-zone module entirely. Programmable option for cycle-by-cycle trip clear If desired, independently configure trip actions taken when time-base counter is counting down.
Event Trigger (ET)	<ul style="list-style-type: none"> Enable the ePWM events that will trigger an interrupt. Enable ePWM events that will trigger an ADC start-of-conversion event. Specify the rate at which events cause triggers (every occurrence or every 2nd or up to 15th occurrence) Poll, set, or clear event flags
Digital Compare (DC)	<ul style="list-style-type: none"> Enables comparator (COMP) module outputs and trip zone signals which are configured using the Input X-BAR to create events and filtered events 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.

13.3.2 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 13-4 illustrates the time-base module's place within the ePWM.

Figure 13-4. Time-Base Submodule



13.3.2.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

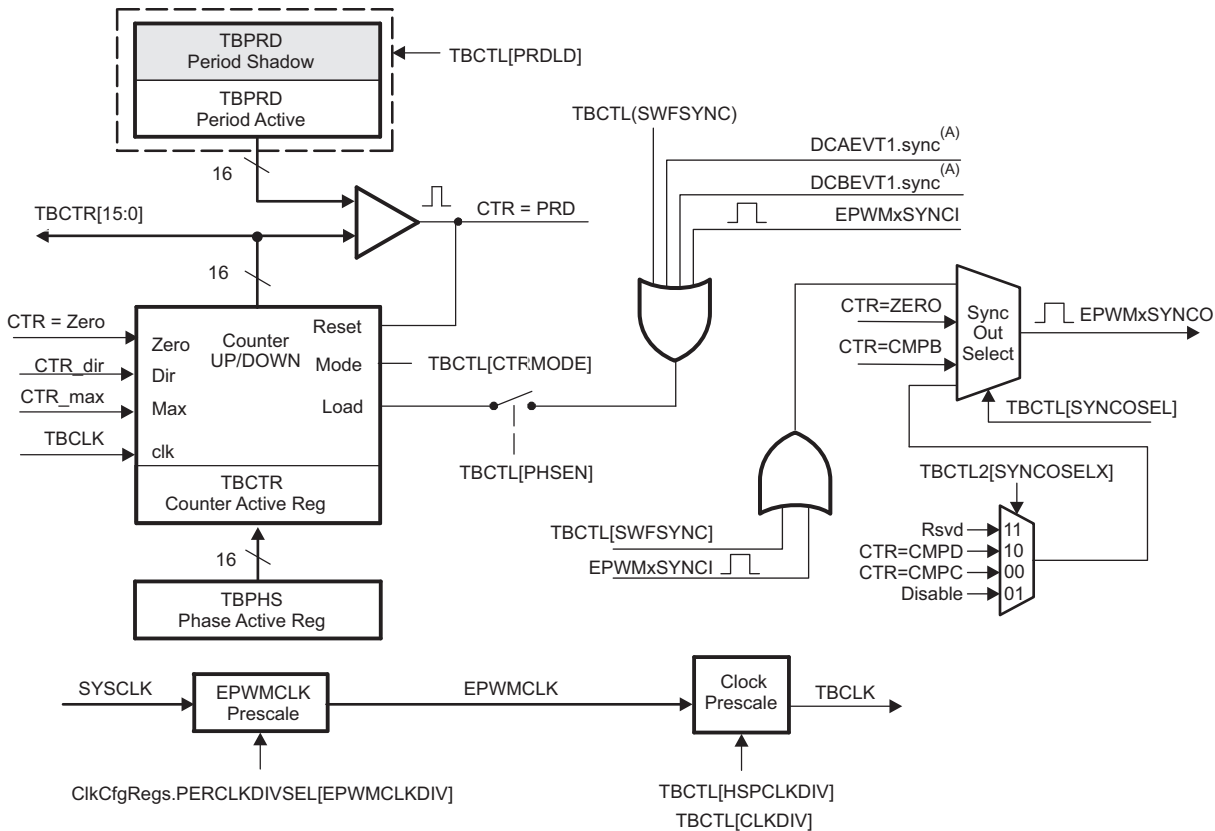
- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate 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.

Note: The Type 4 EPWM clocking varies from previous EPWM types. Prior to the Type 4 EPWM, the time-base submodule was clocked directly by the system clock (SYSCLKOUT). On this version of the ePWM, there is a divider (EPWMCLKDIV) of the system clock which defaults to EPWMCLK = SYSCLKOUT / 2.

13.3.2.2 Controlling and Monitoring the Time-Base Submodule

The block diagram in [Figure 13-5](#) shows the critical signals and registers of the time-base submodule. [Table 13-2](#) provides descriptions of the key signals associated with the time-base submodule.

Figure 13-5. Time-Base Submodule Signals and Registers



A. These signals are generated by the digital compare (DC) submodule.

Table 13-2. Key Time-Base Signals

Signal	Description
EPWMxSYNCl	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, EPWM2SYNCl is generated by the ePWM1 peripheral, EPWM3SYNCl is generated by ePWM2 and so forth. For information on the synchronization order of a particular device, see Section 13.3.2.3.3 .
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: <ol style="list-style-type: none"> 1. EPWMxSYNCl (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.
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.
CTR = Zero	Time-base counter equal to zero This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x00.

Table 13-2. Key Time-Base Signals (continued)

Signal	Description
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.

13.3.2.3 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 13-6](#) 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:**

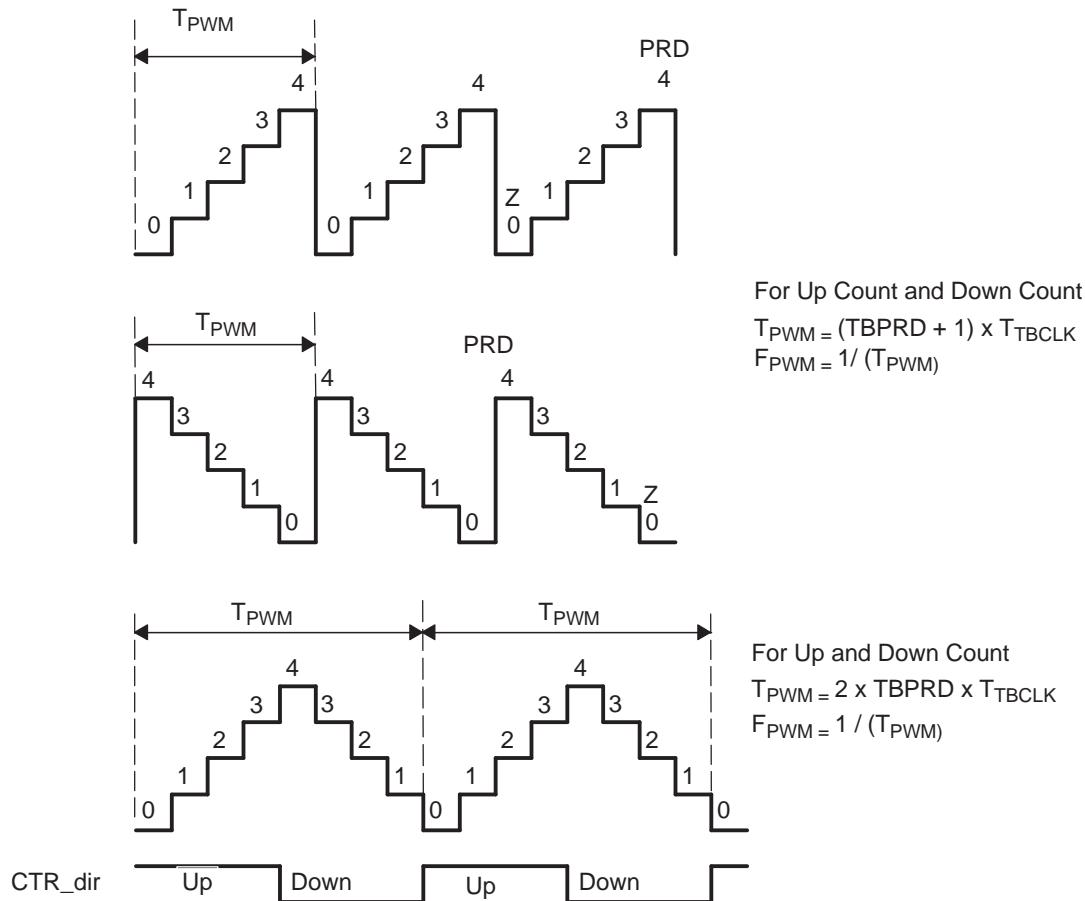
In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.

- **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 13-6. Time-Base Frequency and Period


13.3.2.3.1 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 13.3.2.3.3](#).

The global reload 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 reload 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 reload control mechanism is explained in [Section 13.3.2.7](#).

- **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.

13.3.2.3.2 Time-Base Clock Synchronization

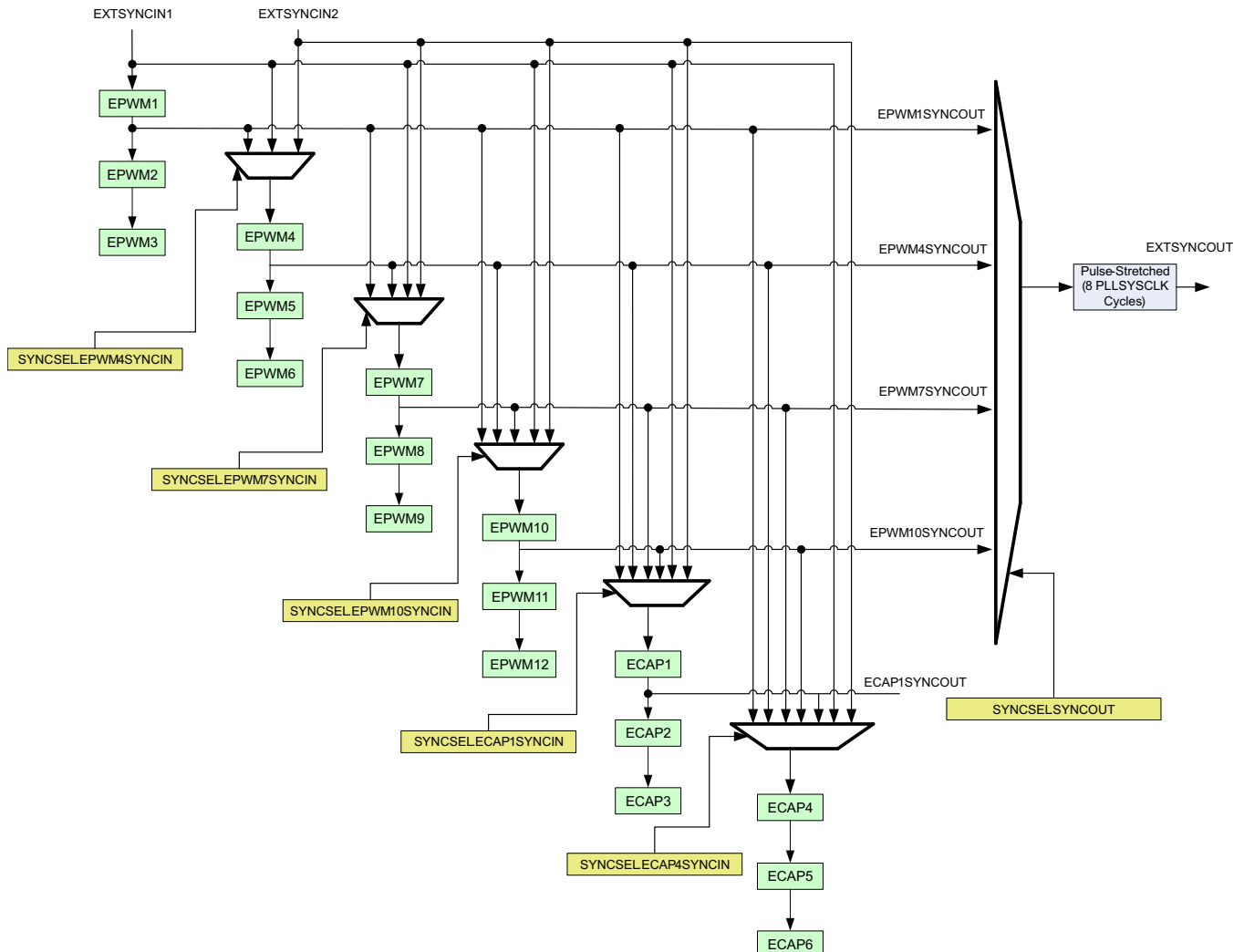
The TBCLKSYNC bit in the peripheral clock enable registers allows all users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescalers 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

13.3.2.3.3 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 (SYNCl) and a synchronization output (SYNCO). In [Figure 15-8](#), EXTSYNCl is sourced from INPUTXBAR5 and EXTSYNCO2 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 13-7. Time-Base Counter Synchronization Scheme 4


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:

- **EPWMxSYNCl: 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 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 EPWMxSYNCl.

- **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 13-8](#) through [Figure 13-11](#) 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 13.4](#) for more details on synchronization strategies.

13.3.2.4 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.

13.3.2.5 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.

NOTE: The ePWM register linking scheme works with the TBPRD:TBPRDHR [Mirrored instance], CMPA:CMPAHR [Mirrored instance], CMPB:CMPBHR [Mirrored instance], CMPC, and CMPD registers present in the upper page offset. Only the instance of these registers in offsets 0x62-0x6B are linked between ePWM modules

A typical example snippet will use the following registers linked between modules

EPwmxRegs.TBPRDHRM2, EPwmxRegs.CMPAM2, EPwmxRegs.CMPBM,
EPwmxRegs.CMPC, EPwmxRegs.CMPD

13.3.2.6 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 13-8. Time-Base Up-Count Mode Waveforms

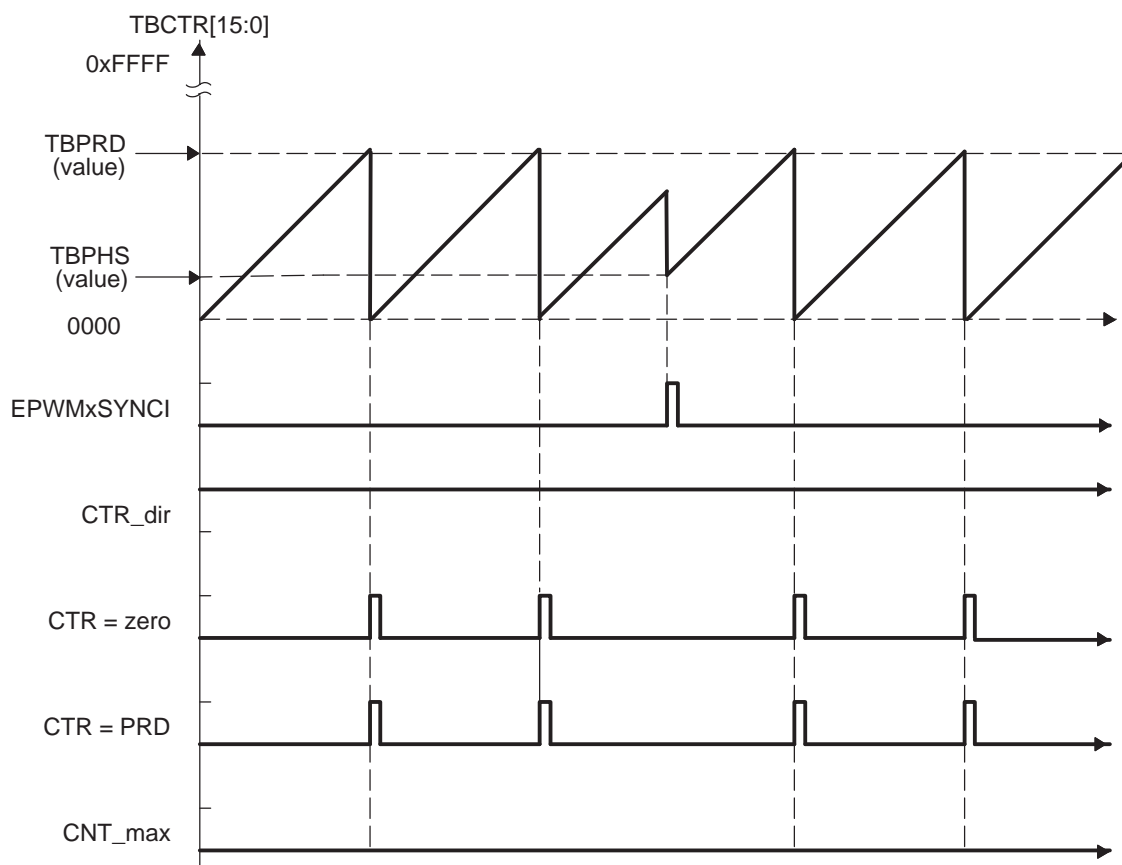


Figure 13-9. Time-Base Down-Count Mode Waveforms

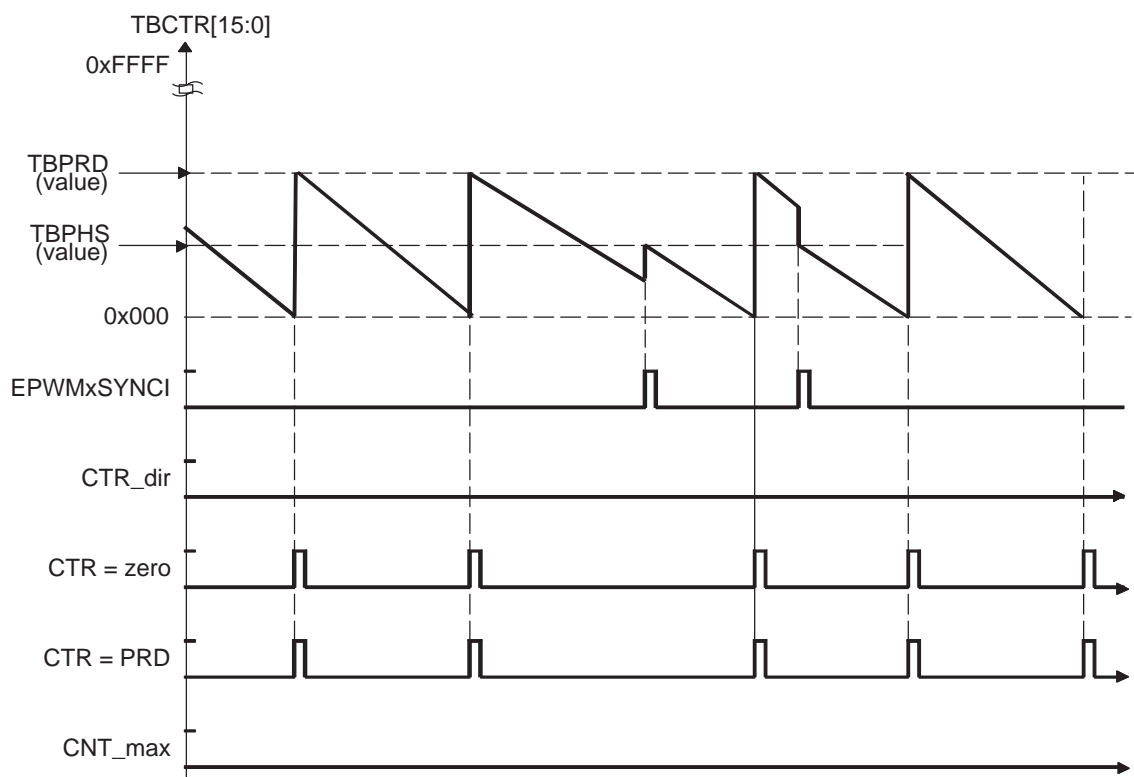


Figure 13-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

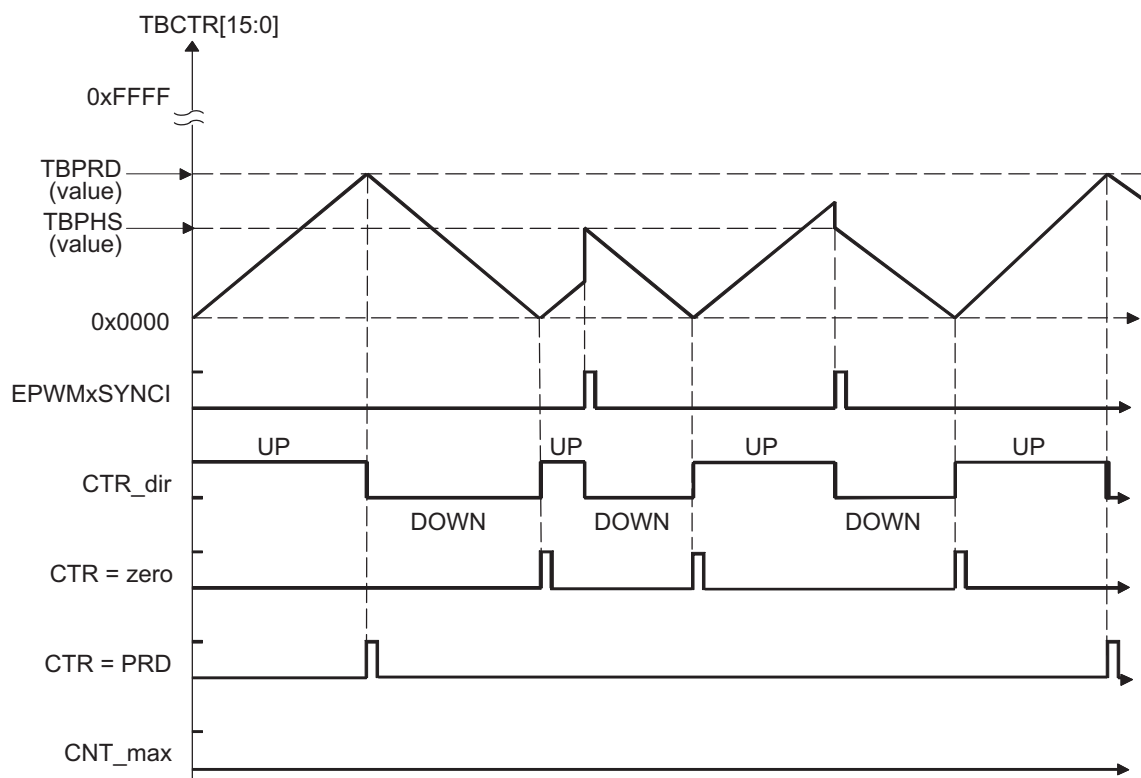
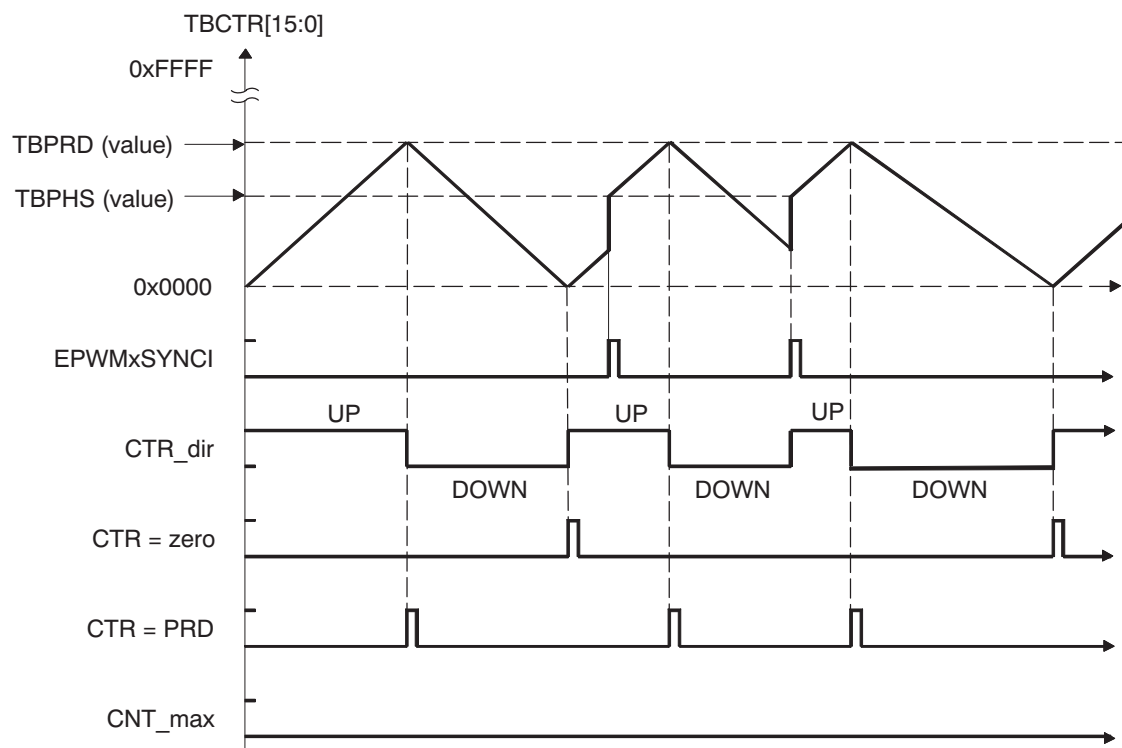


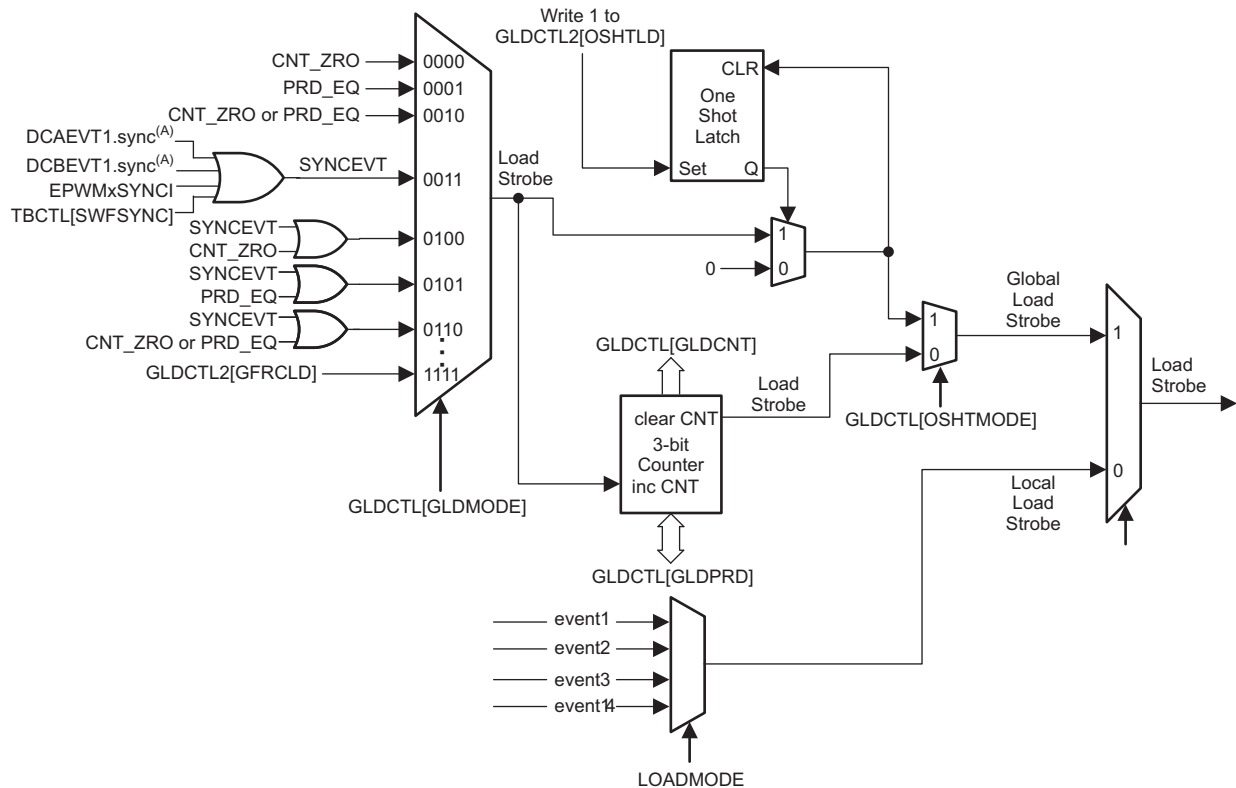
Figure 13-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event



13.3.2.7 Global Reload

Figure 13-12 illustrates the signals and registers associated with the global reload feature.

Figure 13-12. Global Reload: 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.

13.3.2.7.1 Global Reload Pulse Pre-Scalar

This feature provides the capability to choose shadow to active transfers to happen once in 'N' occurrences of selected global reload 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').

13.3.2.7.2 One-Shot Reload Mode

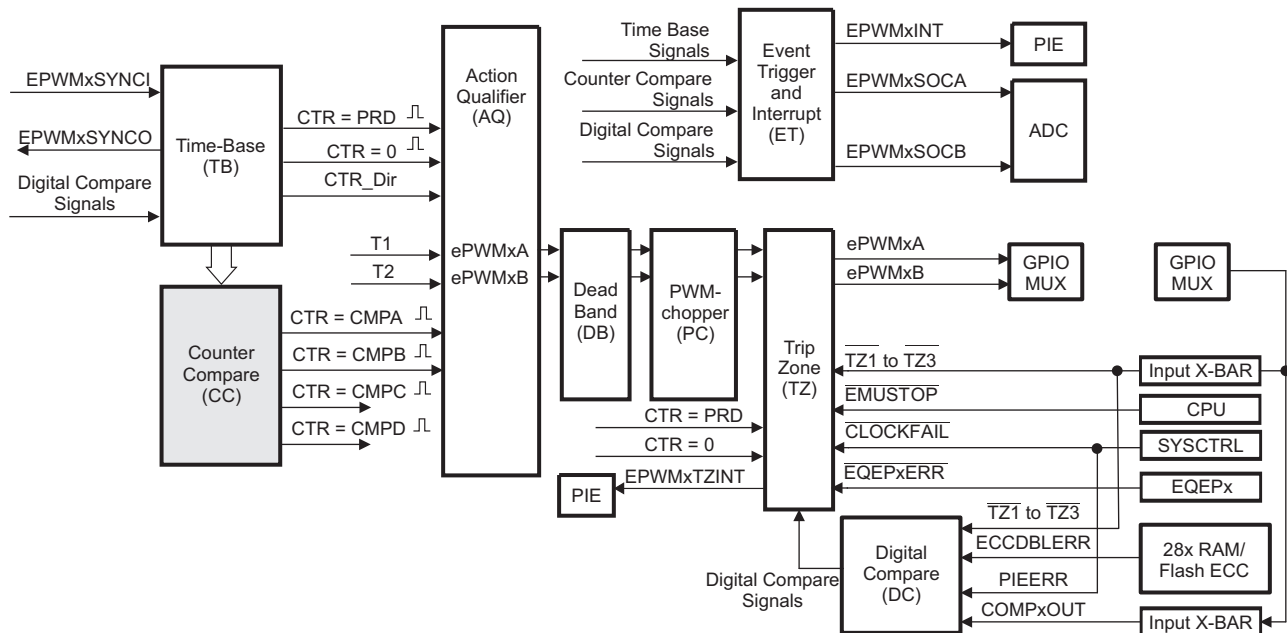
This feature allows users to cause the shadow register to active register transfers to occur once. When GLDCTL2[OSHTLD] = '1' the shadow to active register transfer, for registers that are configured to use the global load mechanism, takes place on the event selected by GLDCTL[GLDMODE].

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 reload 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.

13.3.3 Counter-Compare (CC) Submodule

Figure 13-13 illustrates the counter-compare submodule within the ePWM.

Figure 13-13. Counter-Compare Submodule



13.3.3.1 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.

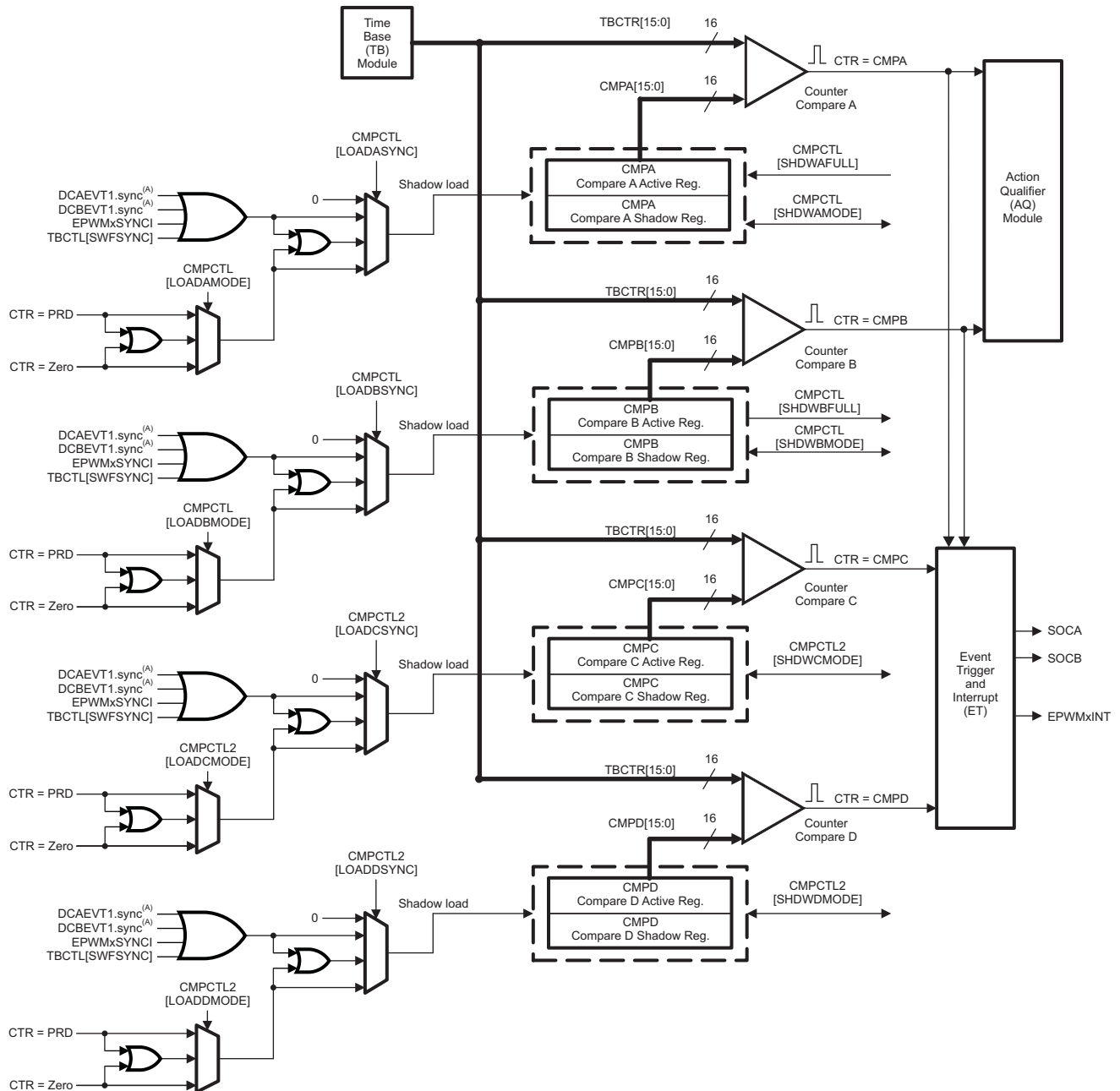
The counter-compare:

- 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

13.3.3.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is shown in Figure 13-14.

Figure 13-14. 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).

13.3.3.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers, which is fed to action-qualifier submodule and event trigger submodule :

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).

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 13.3.4.1](#) 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[LOADDMODE] 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/LOADDMODE

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 Reload Support

The global reload 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 reload 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 reload control mechanism is explained in [Section 13.3.2.7](#).

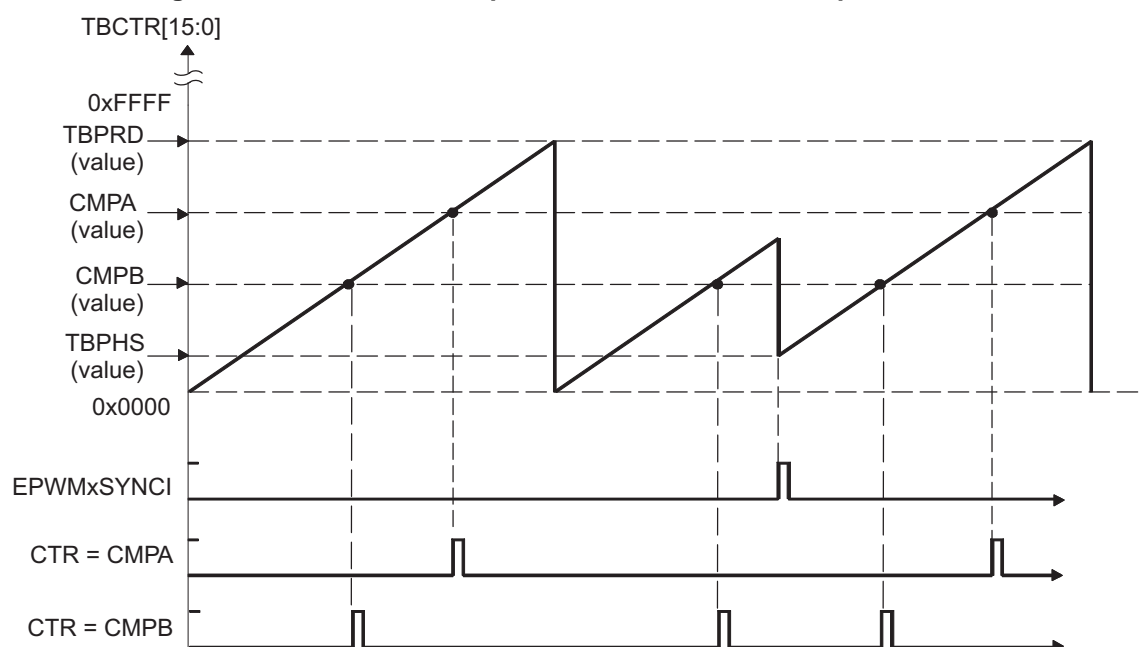
13.3.3.4 Count Mode Timing Waveforms

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 13-15](#) through [Figure 13-18](#) show when events are generated and how the EPWMxSYNCl signal interacts.

Figure 13-15. 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 13-16. Counter-Compare Events in Down-Count Mode

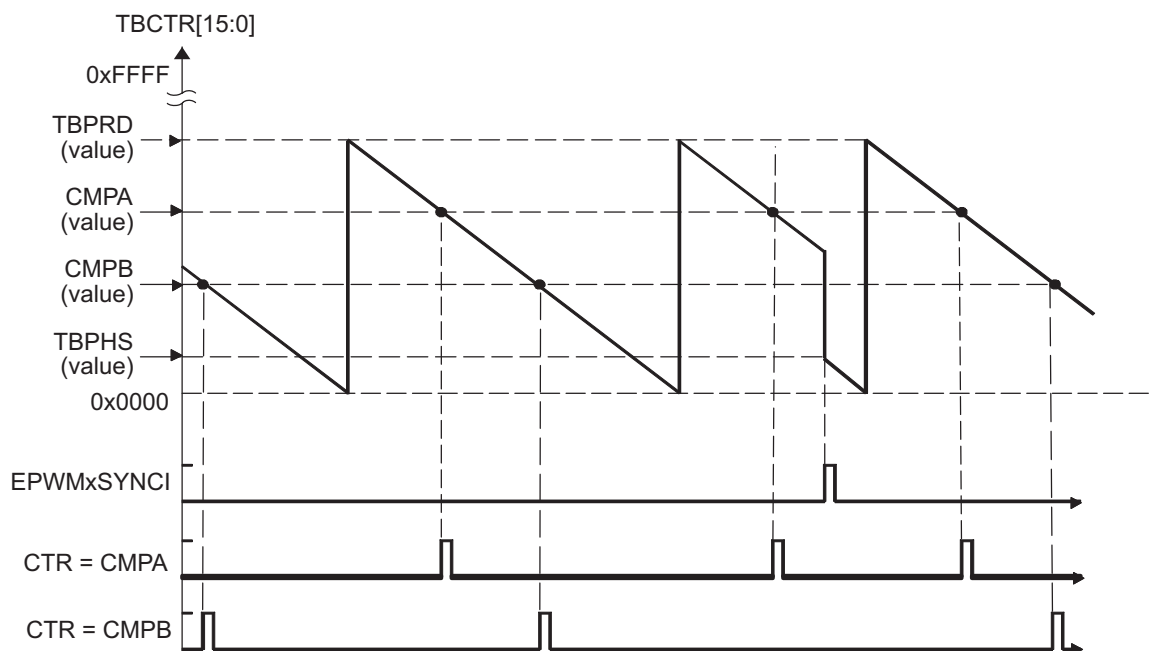


Figure 13-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

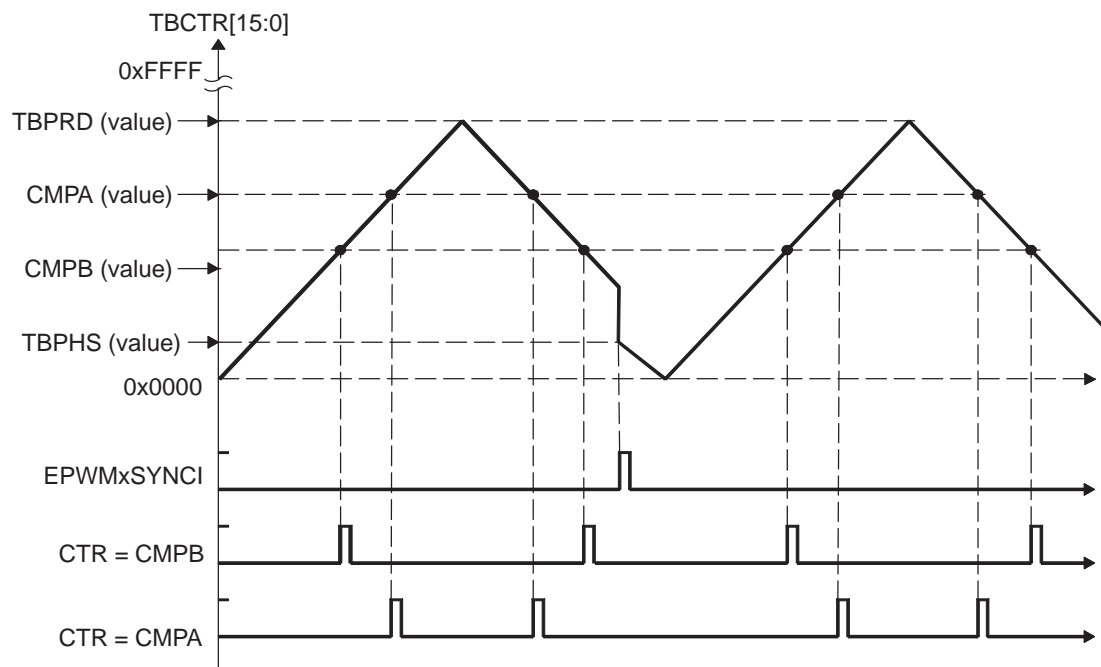
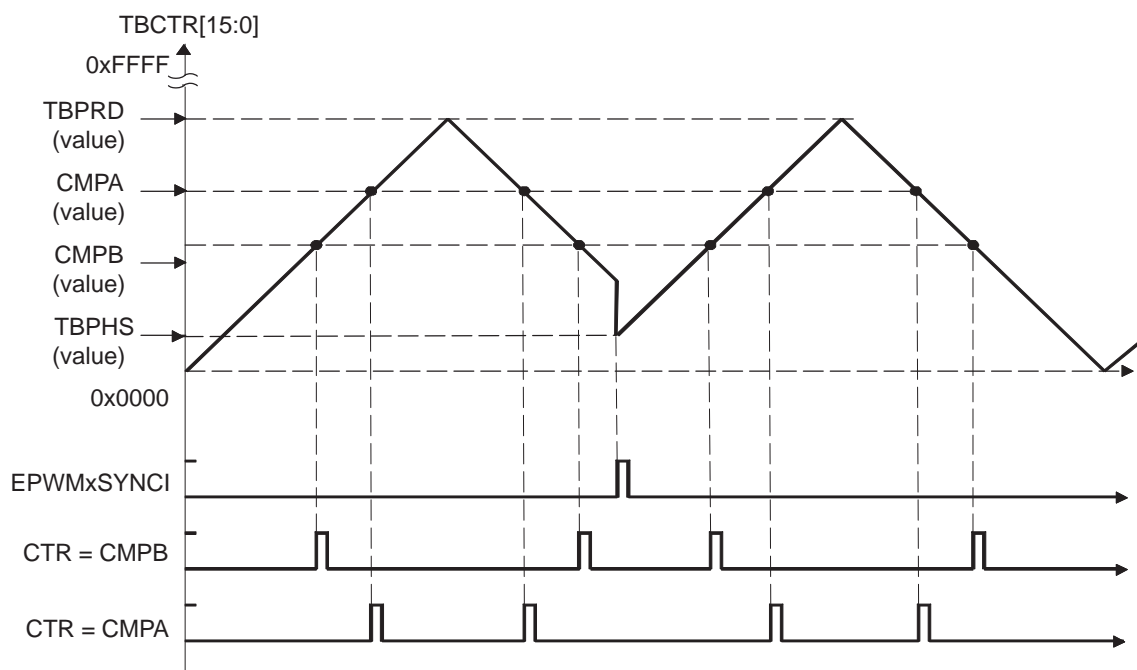
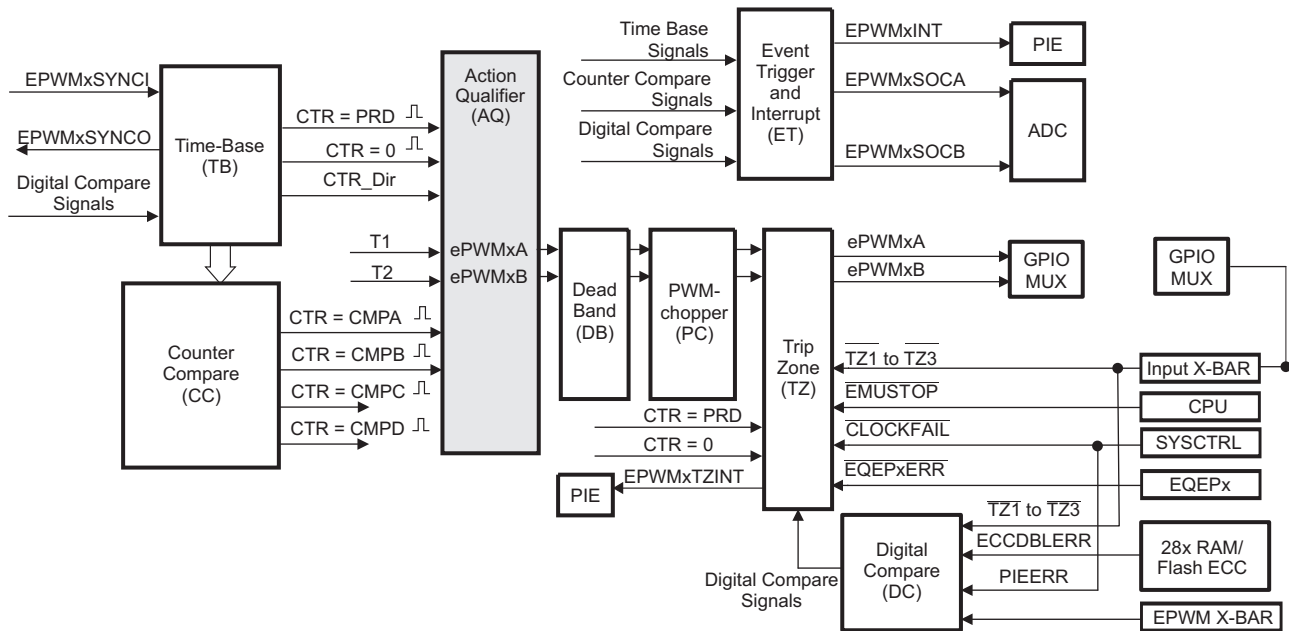


Figure 13-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event



13.3.4 Action-Qualifier (AQ) Submodule

Figure 13-19 shows the action-qualifier (AQ) submodule in the ePWM system.

Figure 13-19. 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.

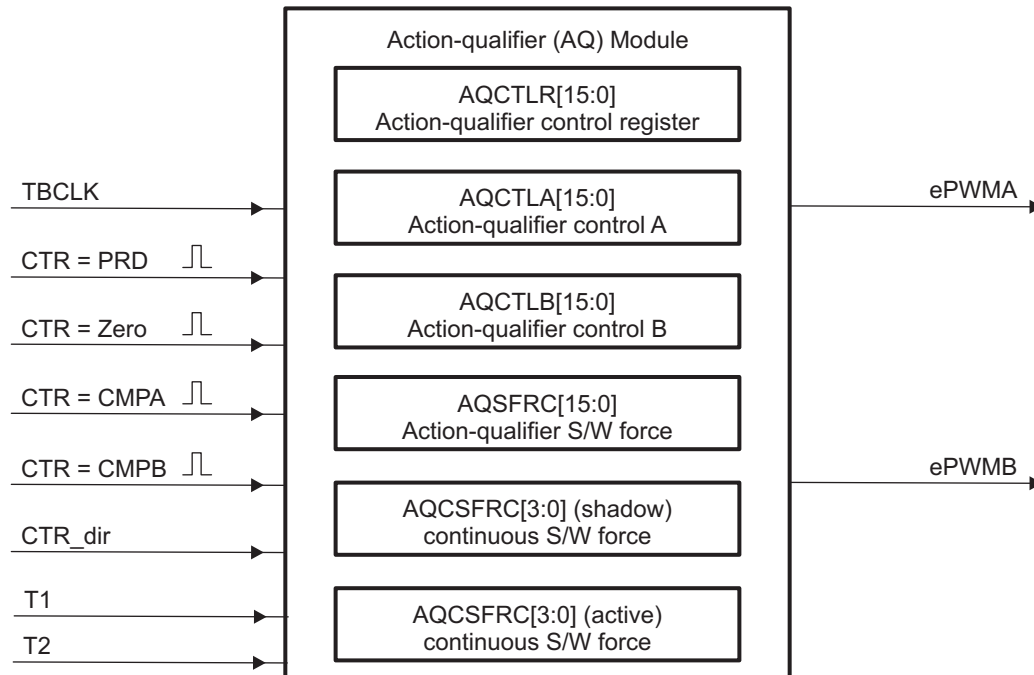
13.3.4.1 Purpose of the Action-Qualifier Submodule

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

13.3.4.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is shown in [Figure 13-20](#) rolled and monitored via the registers in [Section 13.5](#).

Figure 13-20. Action-Qualifier Submodule Inputs and Outputs


For convenience, the possible input events are summarized again in [Table 13-3](#).

Table 13-3. 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 AQSFRC 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 Event-trigger Submodule description in [Section 13.3.8](#) 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 13-21](#). 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 13-21. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs

S/W 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.

13.3.4.3 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 [Table 13-4](#). 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 13-4. 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)

[Table 13-5](#) shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

Table 13-5. 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

[Table 13-6](#) 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 13-6. 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)
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (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 [Table 13-7](#).

Table 13-7. 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	<p>If $CMPA/CMPB \leq TBPRD$ period, then the event occurs on a compare match ($TBCTR = CMPA$ or $CMPB$).</p> <p>If $CMPA/CMPB > TBPRD$, then the event will not occur.</p>	Never occurs.
Down-Count Mode	Never occurs.	<p>If $CMPA/CMPB < TBPRD$, the event will occur on a compare match ($TBCTR = CMPA$ or $CMPB$).</p> <p>If $CMPA/CMPB \geq TBPRD$, the event will occur on a period match ($TBCTR = TBPRD$).</p>
Up-Down-Count Mode	<p>If $CMPA/CMPB < TBPRD$ and the counter is incrementing, the event occurs on a compare match ($TBCTR = CMPA$ or $CMPB$).</p> <p>If $CMPA/CMPB \geq TBPRD$, the event will occur on a period match ($TBCTR = TBPRD$).</p>	<p>If $CMPA/CMPB < TBPRD$ and the counter is decrementing, the event occurs on a compare match ($TBCTR = CMPA$ or $CMPB$).</p> <p>If $CMPA/CMPB \geq TBPRD$, the event occurs on a period match ($TBCTR = TBPRD$).</p>

13.3.4.4 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 AQCTLR[SHDWAQAMODE] and AQCTLR[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 AQCTLR[SHDWAQAMODE] bit, and the shadow register for AQCTLB is enabled by setting the AQCTLR[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 AQCTLR[LDAQAMODE] AQCTLR[LDAQBMODE] AQCTLR[LDAQASYNC] & AQCTLR[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 Reload Support

Global reload 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 reload 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 reload control mechanism is explained in [Section 13.3.2.7](#).

Immediate Load Mode:

If immediate load mode is selected (that is, AQCTLR[SHDWAQAMODE] = 0 or AQCTLR[SHDWAQBMODE] = 0), then a read from or a write to the register will go directly to the active register. See [Figure 13-22](#) and [Figure 13-23](#).

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 AQCTLR 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 13-22. AQCTLR[SHDWAQAMODE]

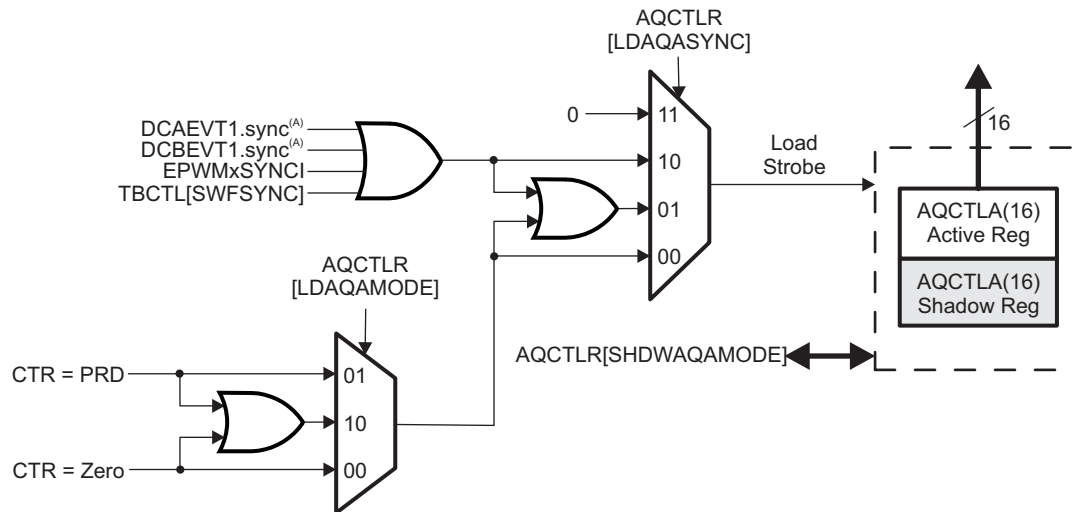
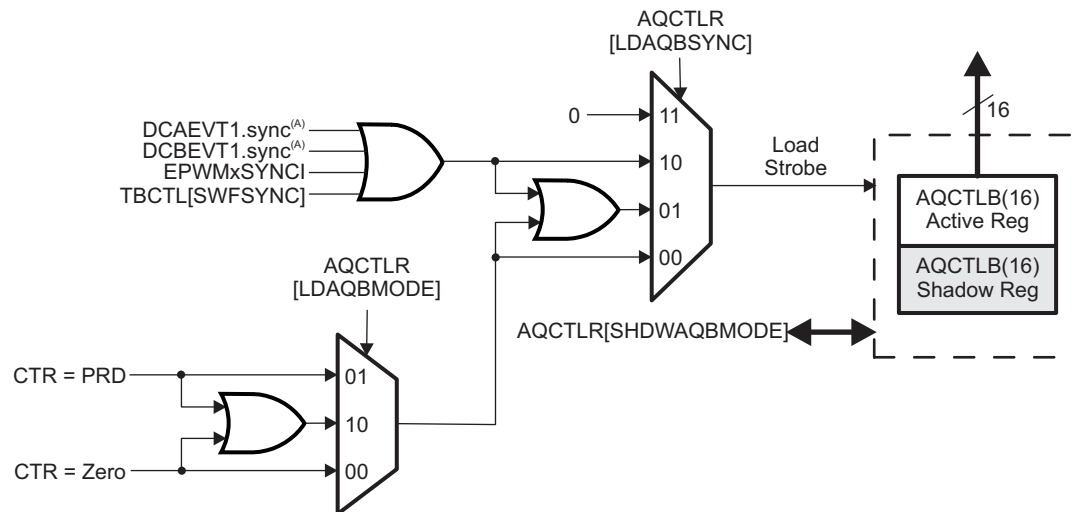


Figure 13-23. AQCTLR[SHDWAQBMODE]



13.3.4.5 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.

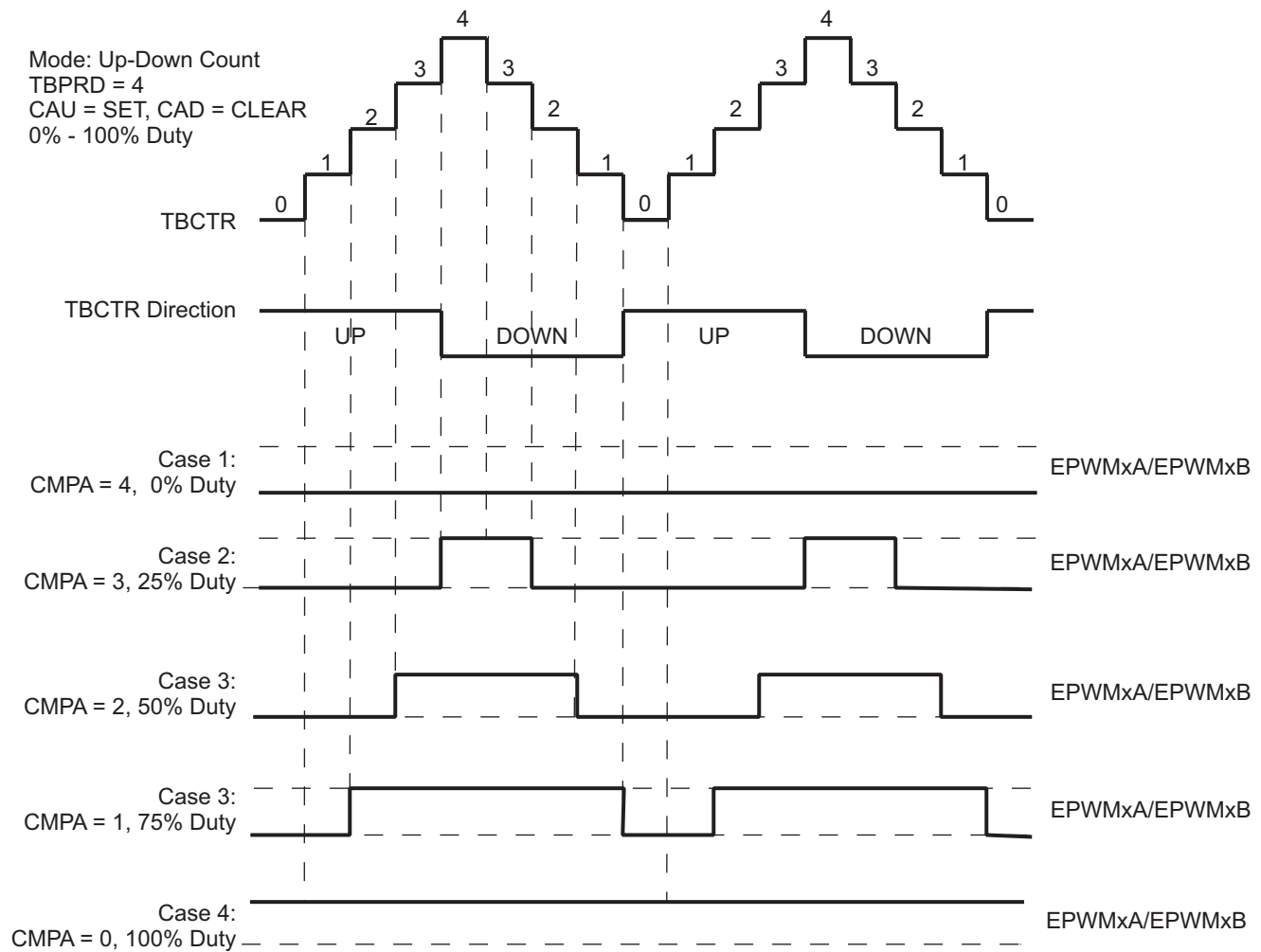
When using up-count mode to generate an asymmetric PWM:

- 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](#))

Figure 13-24 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.

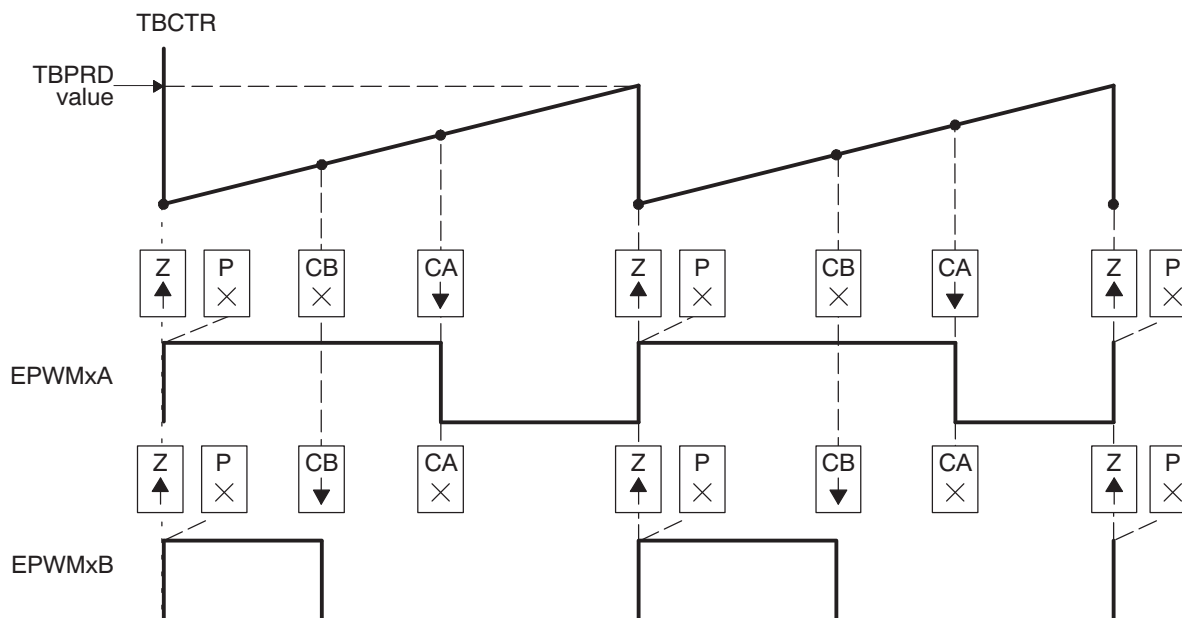
When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Figure 13-24. Up-Down-Count Mode Symmetrical Waveform


The PWM waveforms in [Figure 13-25](#) through [Figure 13-30](#) show some common action-qualifier configurations. Some conventions used in the figures and examples are as follows:

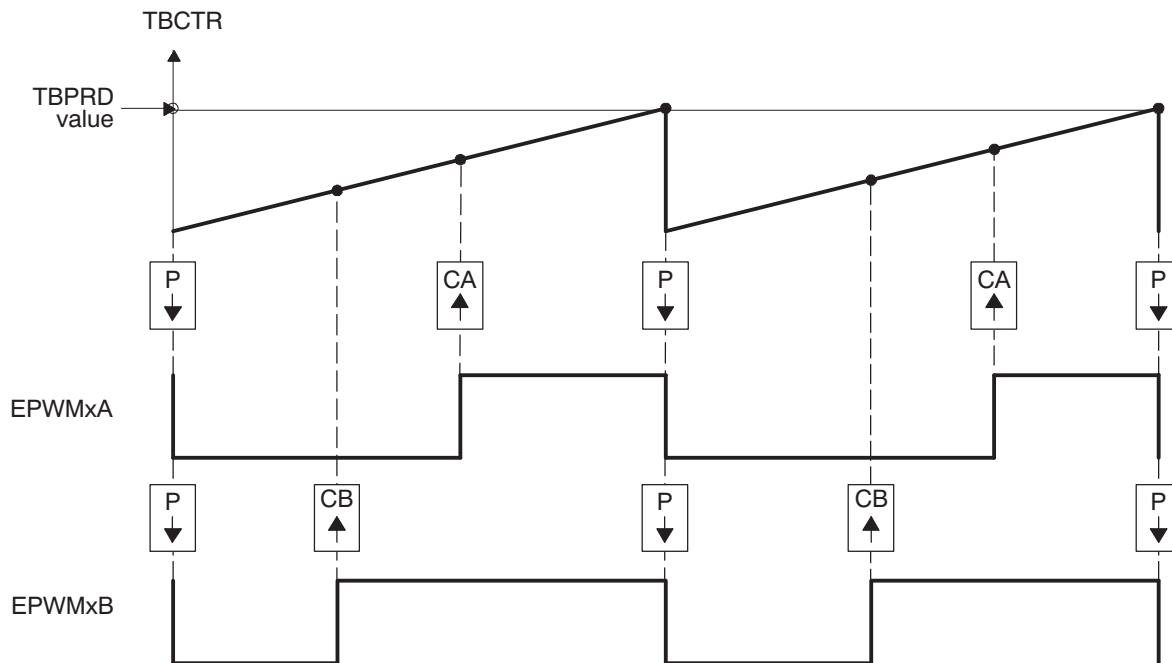
- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

Figure 13-25. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High



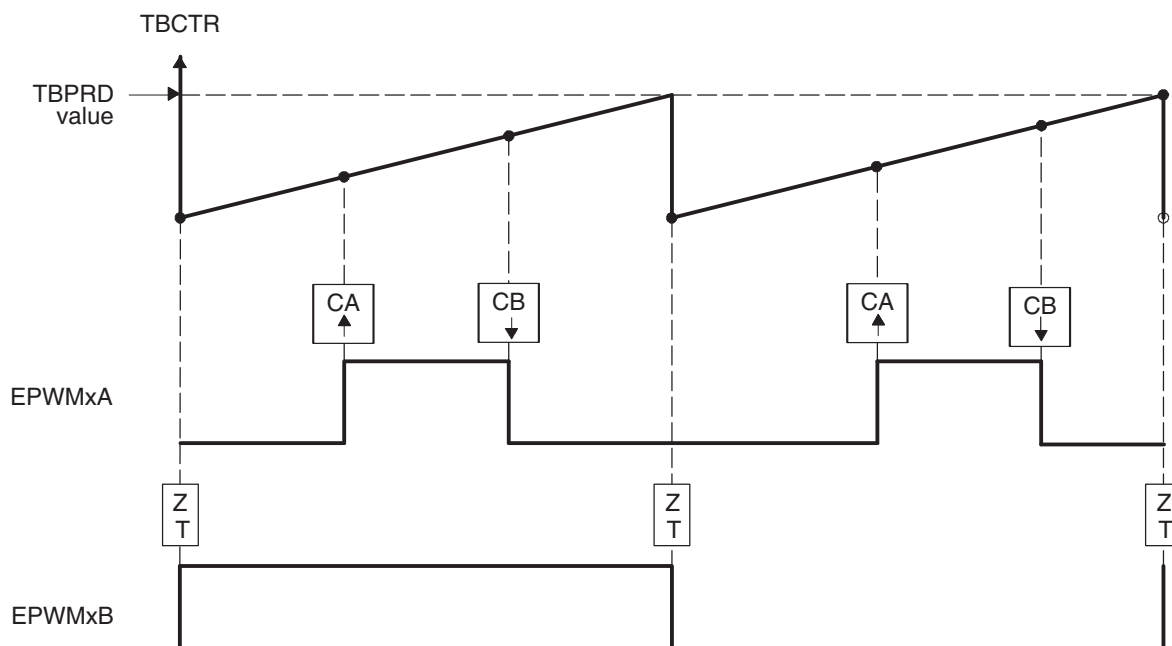
- A PWM period = $(TBPRD + 1) \times T_{TBCLK}$
- B Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
- D The "Do Nothing" actions (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 13-26. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low



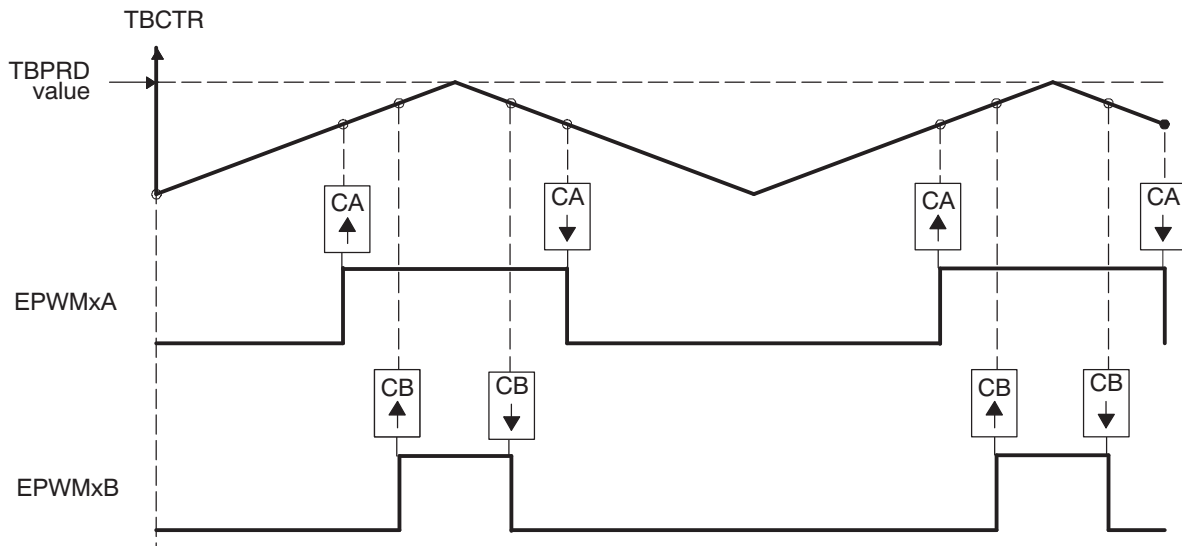
- A PWM period = $(TBPRD + 1) \times T_{TBCLK}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Figure 13-27. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA



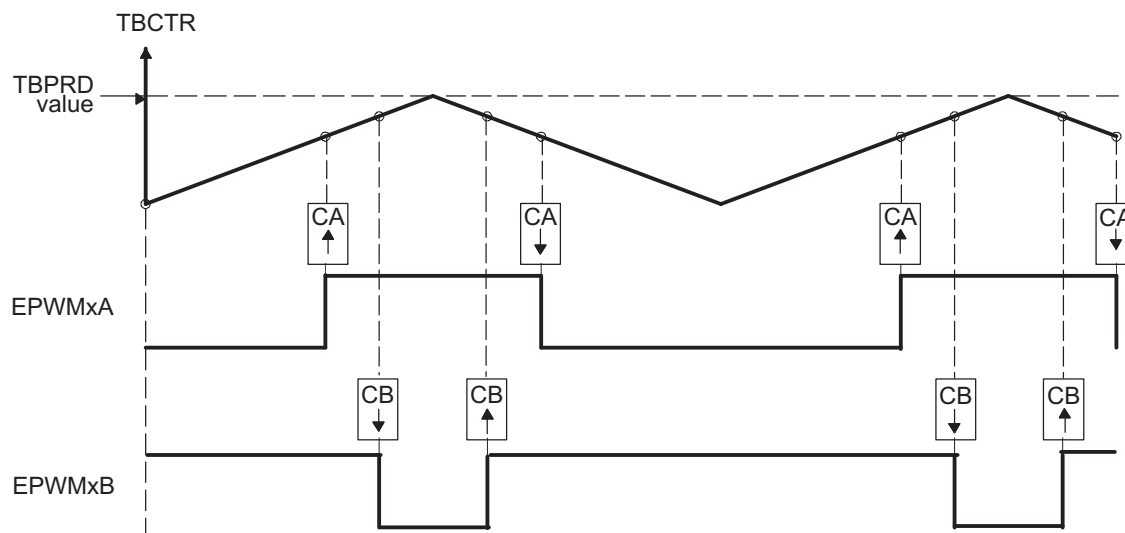
- A PWM frequency = $1 / ((TBPRD + 1) \times T_{TBCLK})$
- B Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)
- C High time duty proportional to (CMPB - CMPA)

Figure 13-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low



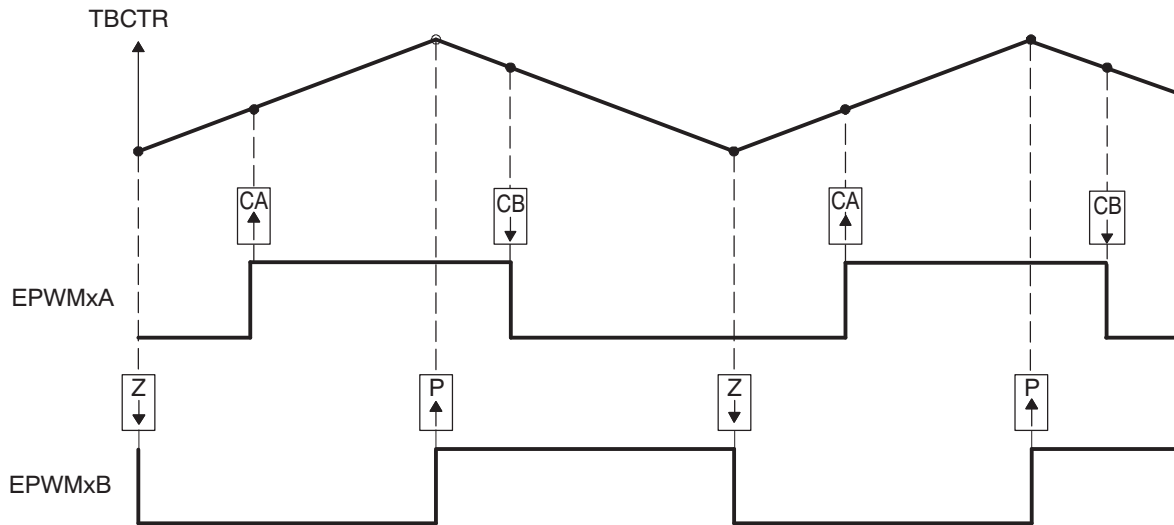
- A PWM period = $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D Outputs EPWMxA and EPWMxB can drive independent power switches.

Figure 13-29. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary



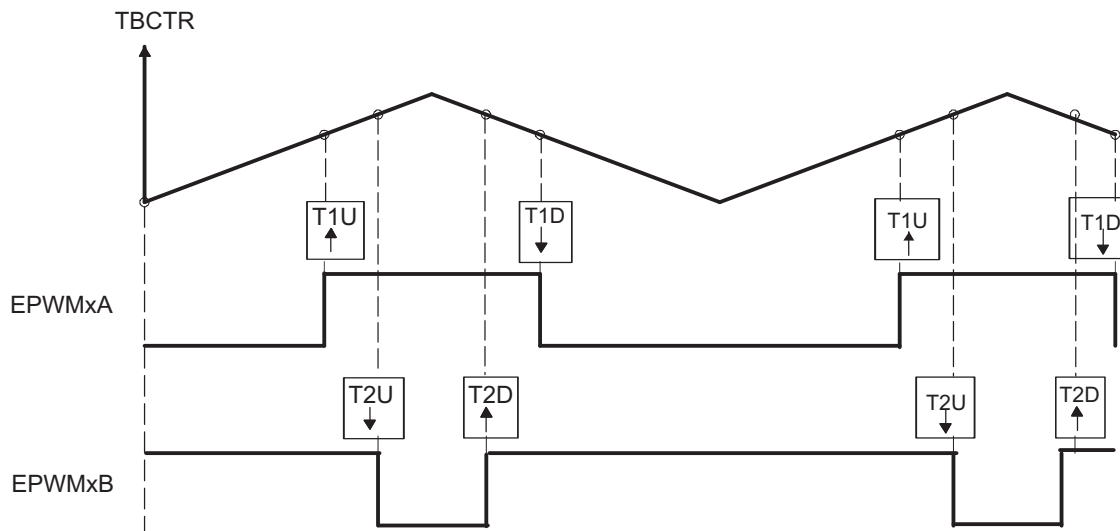
- A PWM period = $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
- C Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
- D Outputs EPWMx can drive upper/lower (complementary) power switches.
- E Dead-band = CMPB - 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 13-30. 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 (CMPA + 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 13-31. Up-Down-Count, PWM Waveform Generation Utilizing T1 and T2 Events

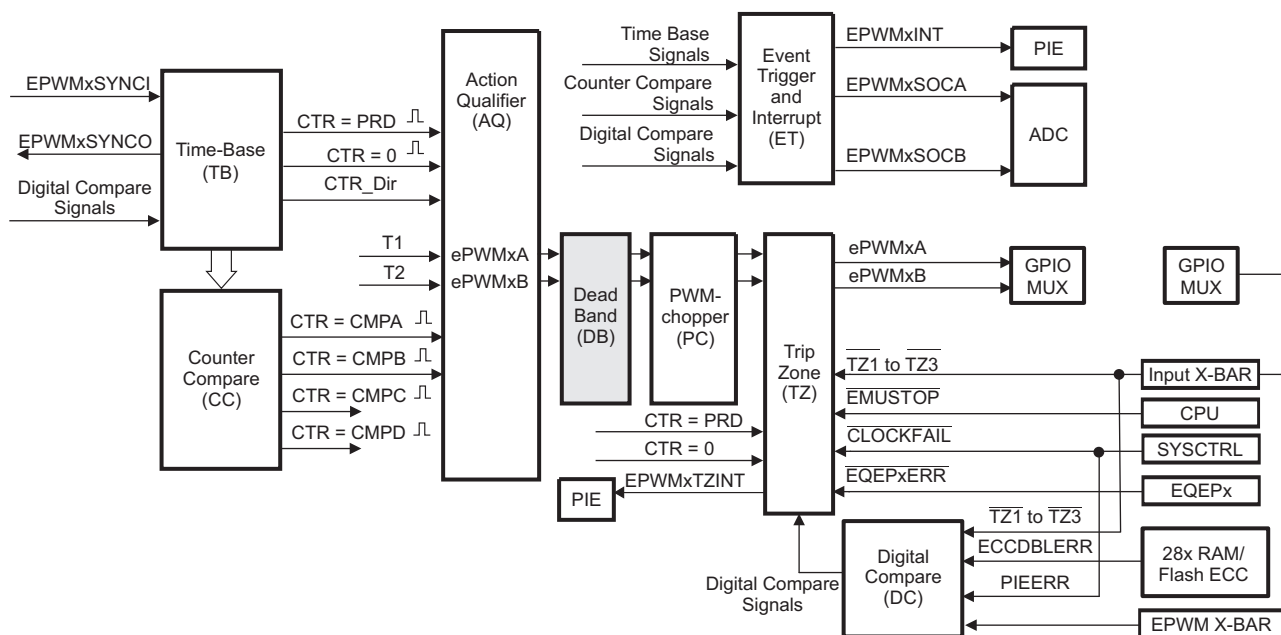


- A PWM period = $2 \times \text{TBPRD} \times \text{TTBCLK}$
- B Independent T1 event actions when counter is counting up and when it is counting down are used to generate EPWMxA output.
- C Independent T2 event actions when counter is counting up and when it is counting down are used to generate EPWMxB output.
- D TZ1 is selected as the source for T1.
- E TZ2 is selected as the source for T2.

13.3.5 Dead-Band Generator (DB) Submodule

Figure 13-32 illustrates the dead-band submodule within the ePWM module.

Figure 13-32. Dead_Band Submodule



13.3.5.1 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)

13.3.5.2 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 13-33, 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.

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 Reload Support

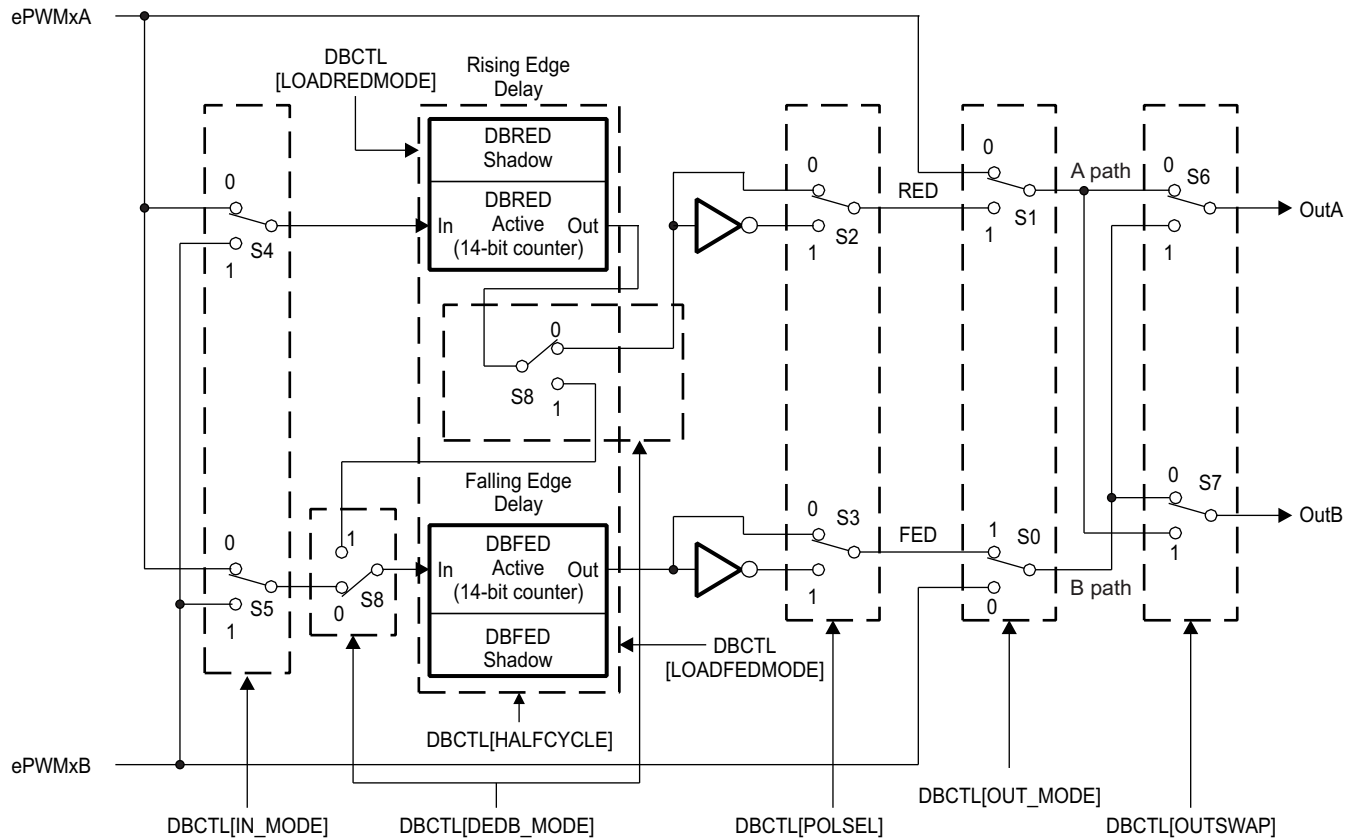
Global reload control mechanism can also be used for DBRED:DBREDHR, DBFED:DBFEDHR and DBCTL registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global reload mode is selected the transfer of contents from shadow register to active register, for all registers 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 reload control mechanism is explained in [Section 13.3.2.7](#).

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.

13.3.5.3 Operational Highlights for the Dead-Band Submodule

The configuration options for the dead-band submodule are shown in [Figure 13-33](#).

Figure 13-33. Configuration Options for the Dead-Band Submodule



Although all combinations are supported, not all are typical usage modes. [Table 13-8](#) 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 13-8](#) 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 13-34](#). Note that to generate equivalent waveforms to [Figure 13-34](#), 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 13-8](#) show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 13-8. 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

Table 13-8. Classical Dead-Band Operating Modes (continued)

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
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				

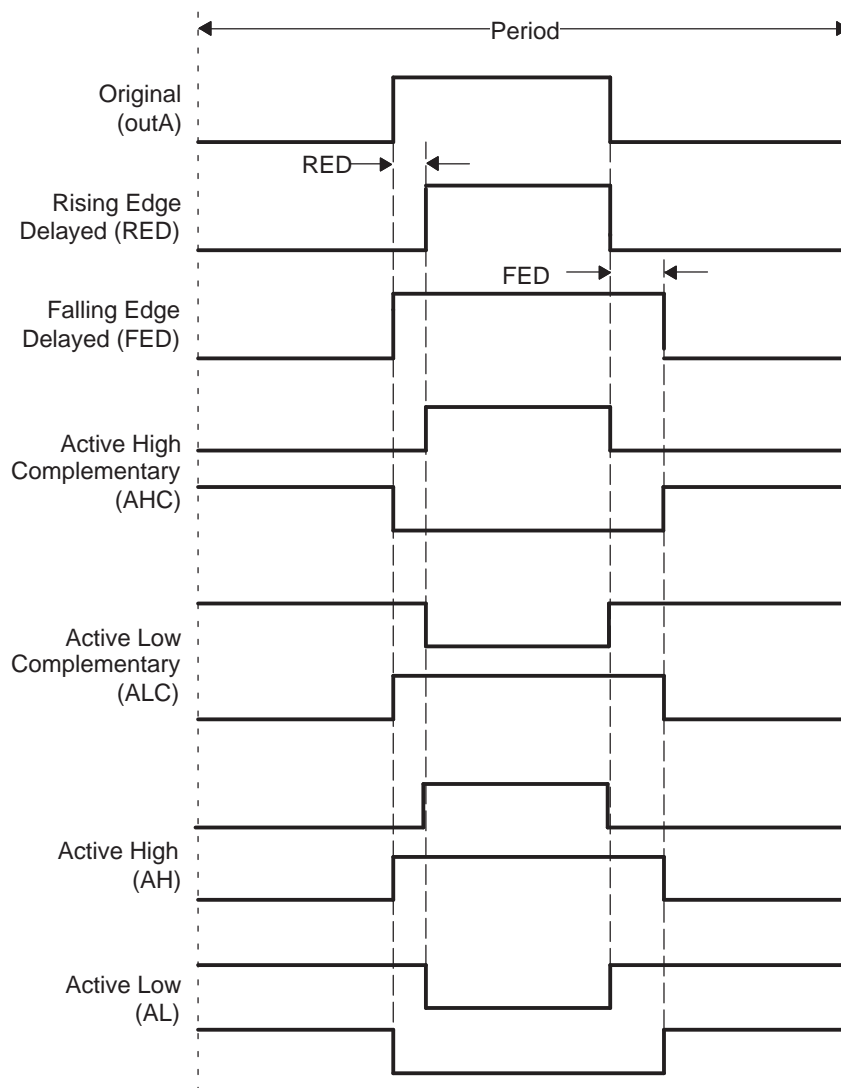
Table 13-9. Additional Dead-Band Operating Modes

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. ⁽¹⁾	1	X	X

⁽¹⁾ 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 13-34 shows waveforms for typical cases where $0\% < \text{duty} < 100\%$.

Figure 13-34. 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:

$$\text{FED} = \text{DBFED} \times T_{\text{TBCLK}}$$

$$\text{RED} = \text{DBRED} \times T_{\text{TBCLK}}$$

Where T_{TBCLK} is the period of TBCLK, the prescaled version of EPWMCLK.

For convenience, delay values for various TBCLK options are shown in [Table 13-10](#). The ePWM input clock frequency that these delay values been computed by is 100 MHz.

Table 13-10. Dead-Band Delay Values in μS as a Function of DBFED and DBRED

Dead-Band Value	Dead-Band Delay in μS		
DBFED, DBRED	TBCLK = EPWMCLK/1	TBCLK = EPWMCLK /2	TBCLK = EPWMCLK/4
1	0.01 μS	0.03 μS	0.05 μS
5	0.06 μS	0.13 μS	0.25 μS
10	0.13 μS	0.25 μS	0.50 μS
100	1.25 μS	2.50 μS	5.00 μS
200	2.50 μS	5.00 μS	10.00 μS
400	5.00 μS	10.00 μS	20.00 μS
500	6.25 μS	12.50 μS	25.00 μS
600	7.50 μS	15.00 μS	30.00 μS
700	8.75 μS	17.50 μS	35.00 μS
800	10.00 μS	20.00 μS	40.00 μS
900	11.25 μS	22.50 μS	45.00 μS
1000	12.50 μS	25.00 μS	50.00 μS

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

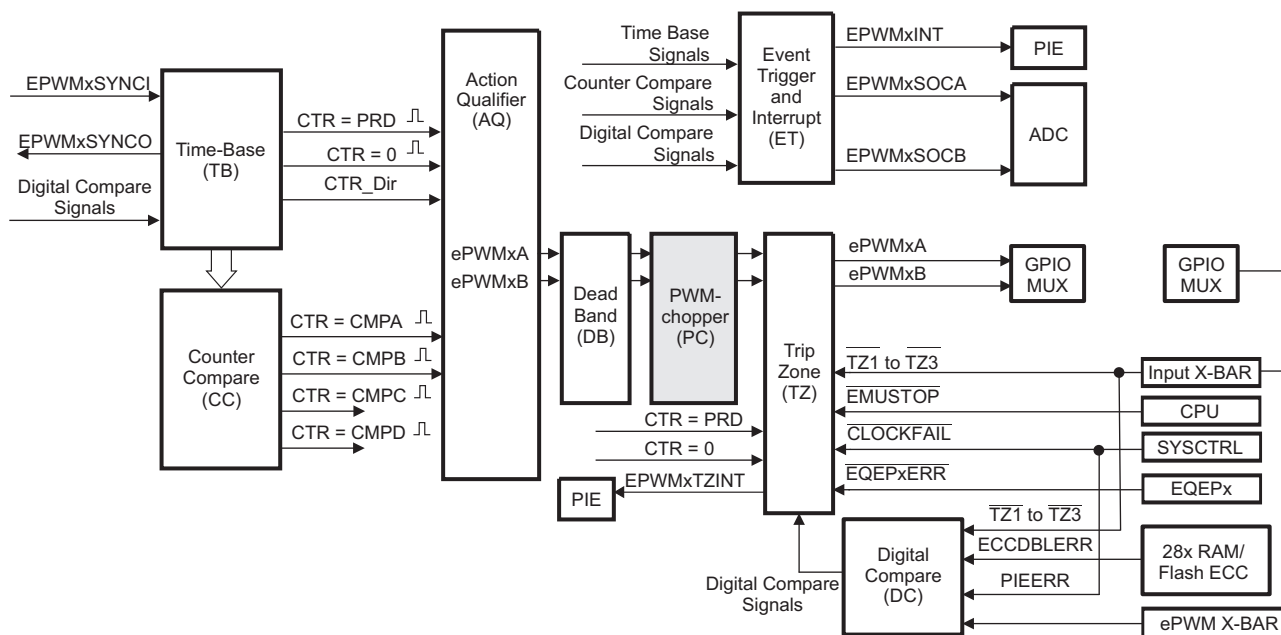
$$\text{FED} = \text{DBFED} \times T_{\text{TBCLK}}/2$$

$$\text{RED} = \text{DBRED} \times T_{\text{TBCLK}}/2$$

13.3.6 PWM Chopper (PC) Submodule

Figure 13-35 illustrates the PWM chopper (PC) submodule within the ePWM module.

Figure 13-35. 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.

13.3.6.1 Purpose of the PWM Chopper Submodule

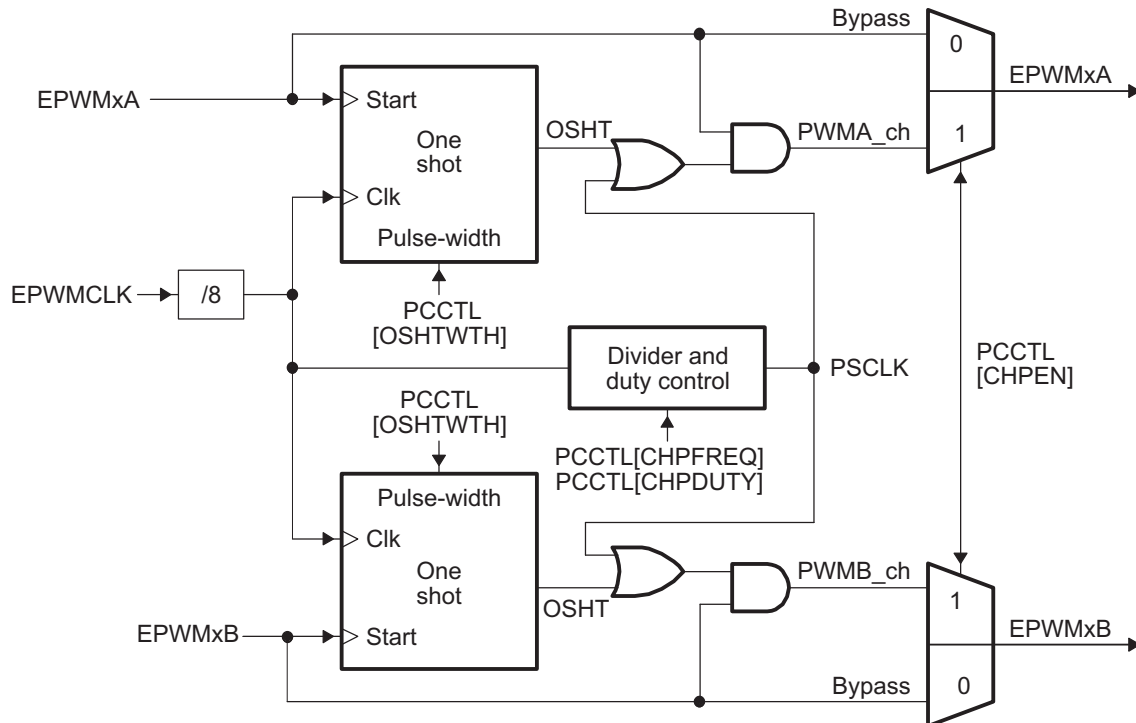
The key functions of the PWM chopper submodule are:

- 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

13.3.6.2 Operational Highlights for the PWM Chopper Submodule

Figure 13-36 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.

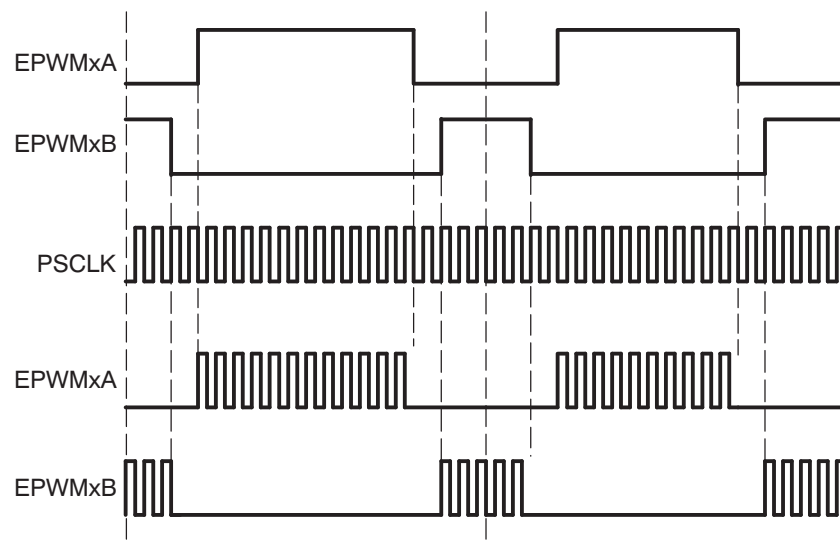
Figure 13-36. PWM Chopper Submodule Operational Details



13.3.6.3 Waveforms

Figure 13-37 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 13-37. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only



13.3.6.3.1 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:

$$T_{1\text{stpulse}} = T_{\text{EPWMCLK}} \times 8 \times \text{OSHTWTH}$$

Where T_{EPWMCLK} is the period of the system clock (EPWMCLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 13-38 shows the first and subsequent sustaining pulses and Table 13-11 gives the possible pulse width values for a EPWMCLK = 80 MHz.

Figure 13-38. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses

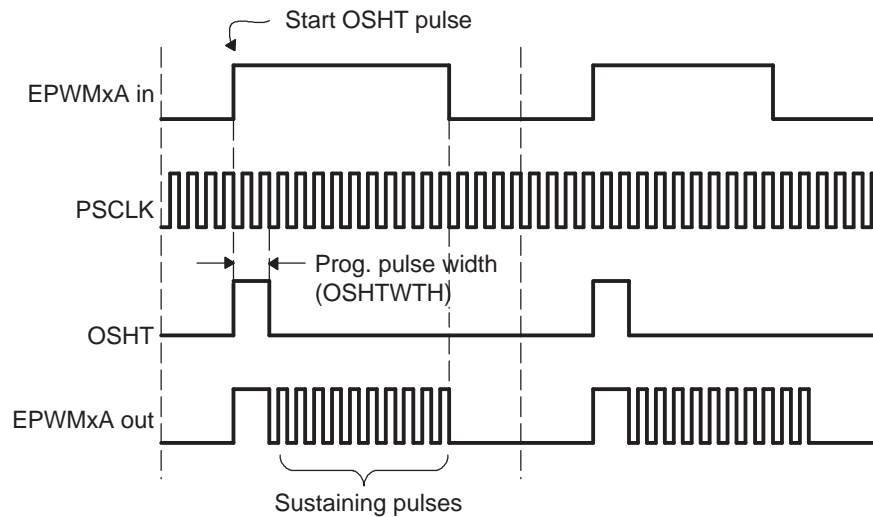


Table 13-11. Possible Pulse Width Values for EPWMCLK = 80 MHz

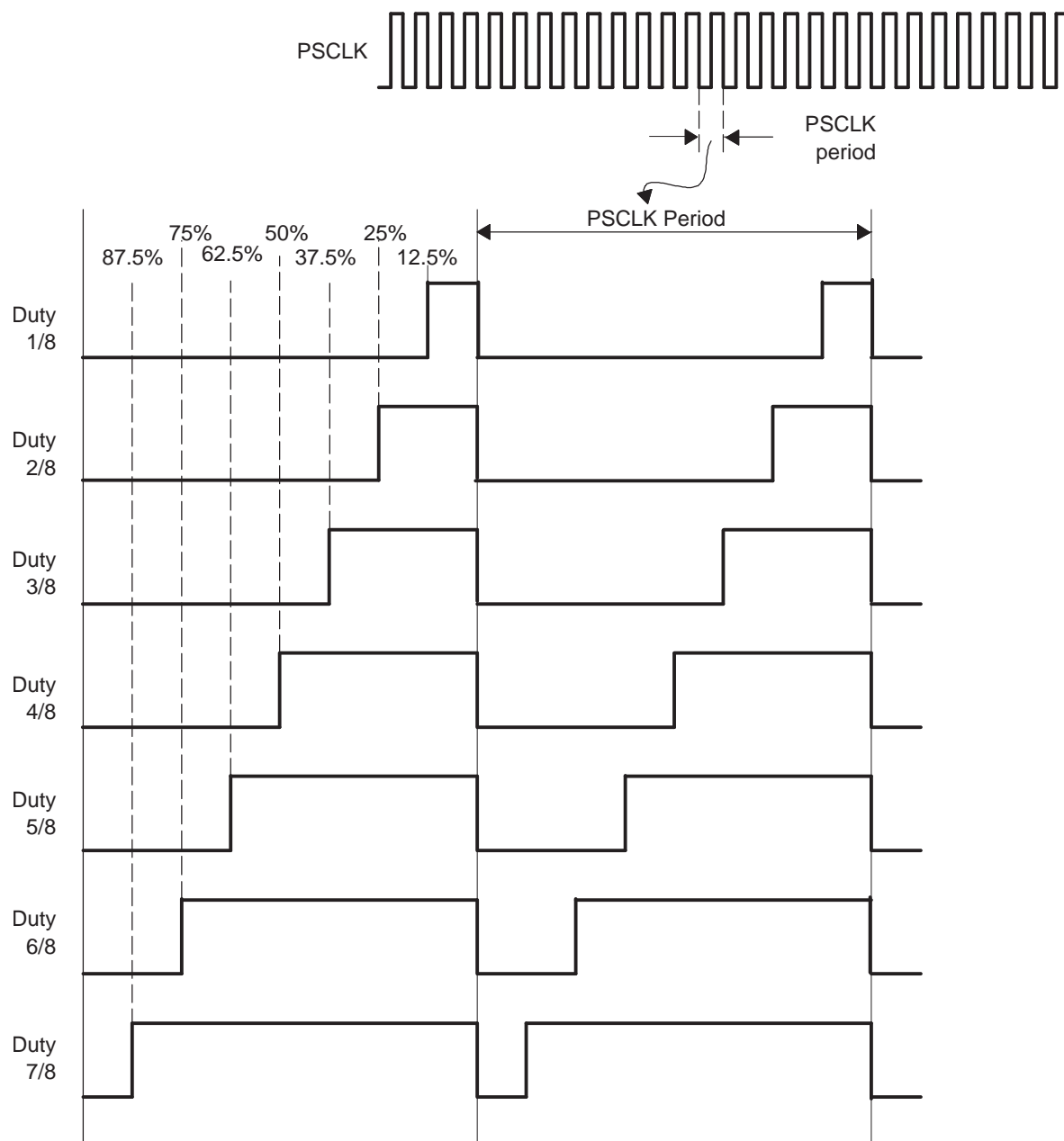
OSHTWTHz (hex)	Pulse Width (nS)
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	900
9	1000
A	1100
B	1200
C	1300
D	1400
E	1500
F	1600

13.3.6.3.2 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 13-39 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%.

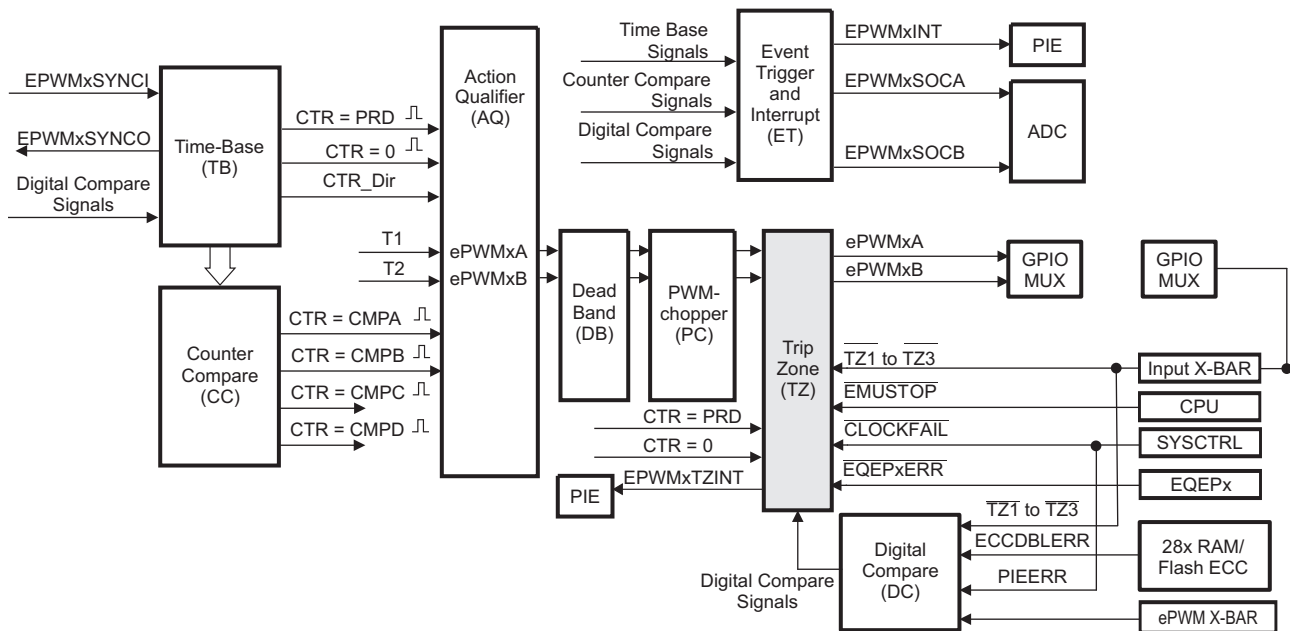
Figure 13-39. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses



13.3.7 Trip-Zone (TZ) Submodule

Figure 13-40 shows how the trip-zone (TZ) submodule fits within the ePWM module.

Figure 13-40. 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.

13.3.7.1 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.

13.3.7.2 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 EPWM clock (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 13-12](#) 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 13-12](#) 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 DCTRISEL 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 13.3.9](#).

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 13-12](#) 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 TZCTLDCA register. Actions specified in TZCTLDCA and TZCTLDCA 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 TZCTLDCA register bit fields. Some of the possible actions, shown in [Table 13-12](#), can be taken on a trip event.

Table 13-12. 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 13-1. 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.
- Configure the ePWM2 registers as follows:

Example 13-1. Trip-Zone Configurations (continued)

- 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.

13.3.7.3 Generating Trip Event Interrupts

[Figure 13-41](#) and [Figure 13-42](#) illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in [Section 13.3.9](#).

Figure 13-41. Trip-Zone Submodule Mode Control Logic

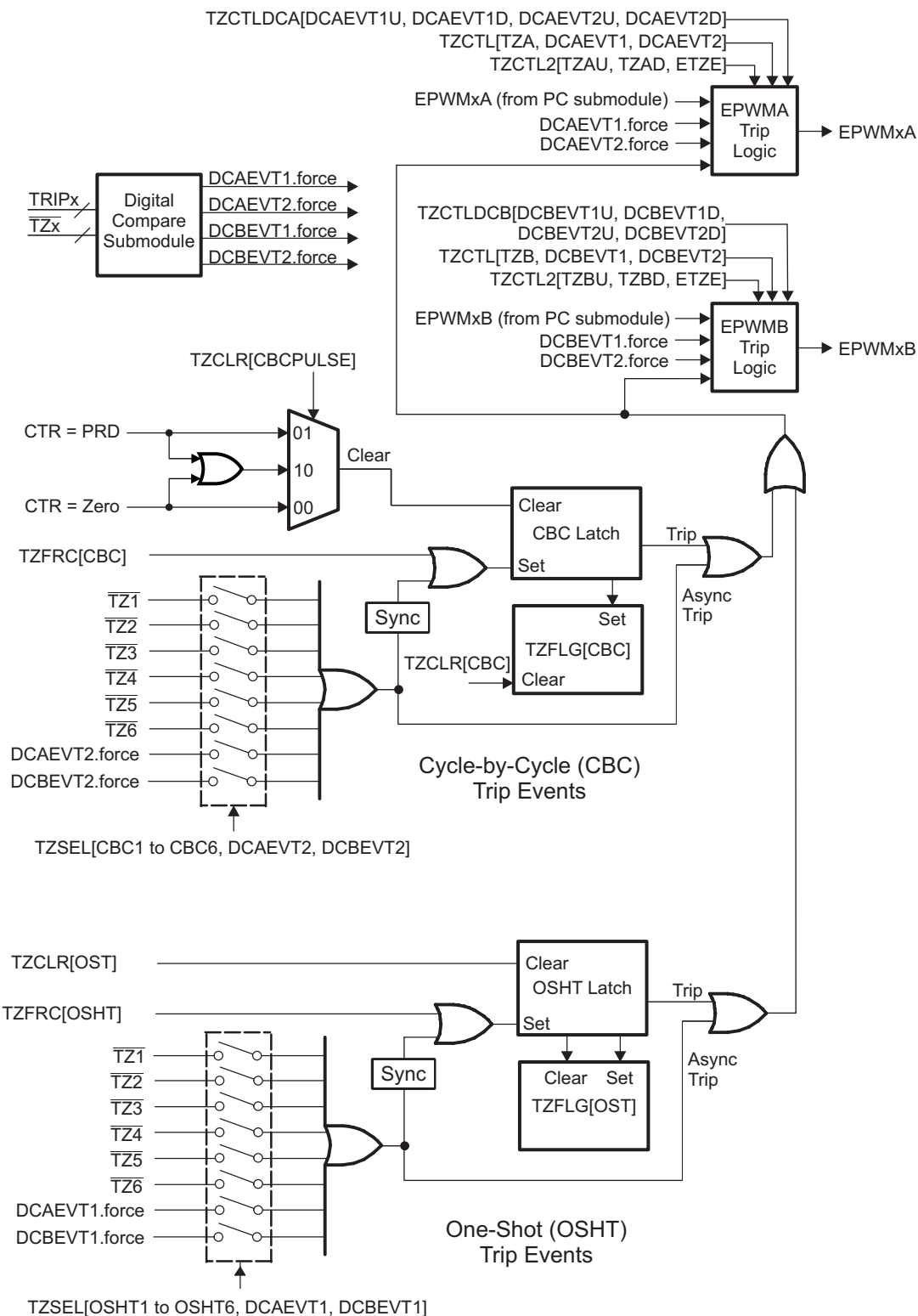
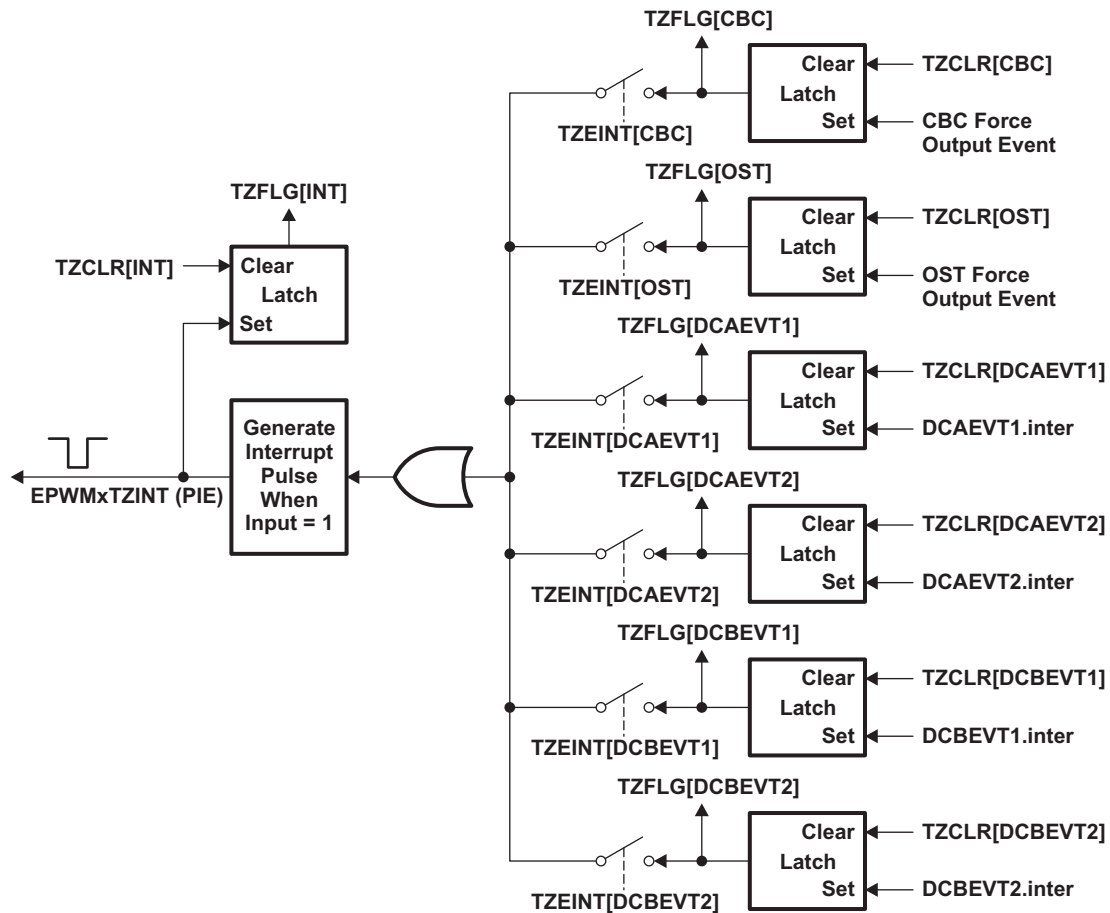


Figure 13-42. Trip-Zone Submodule Interrupt Logic

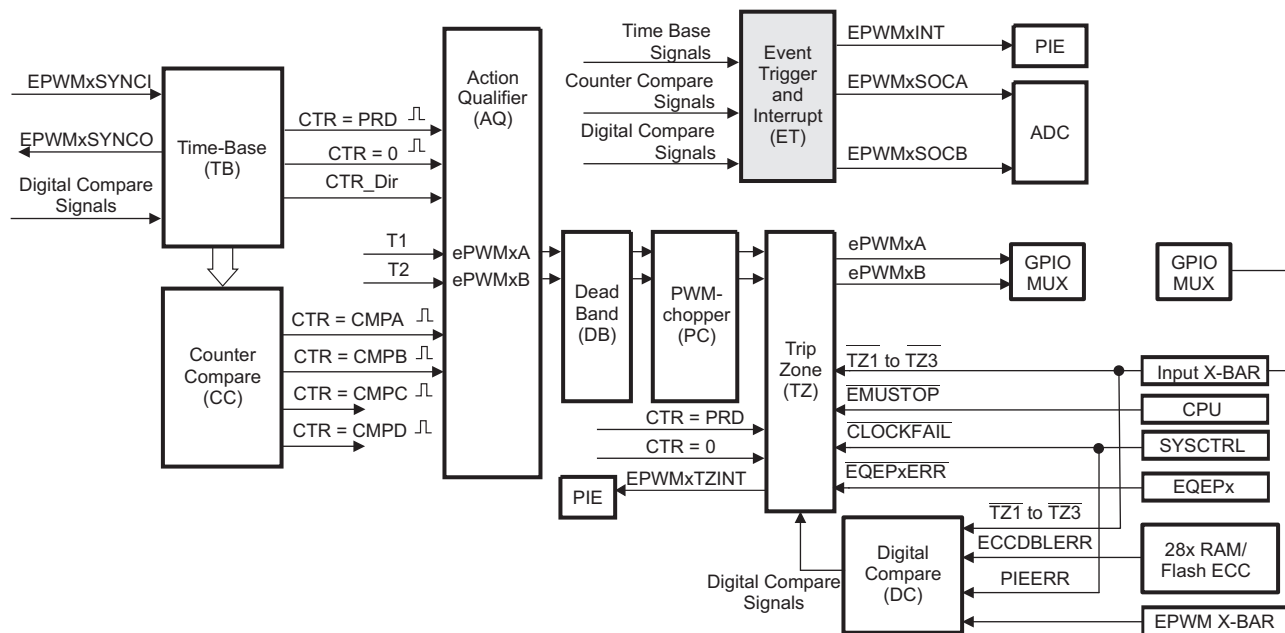


13.3.8 Event-Trigger (ET) Submodule

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

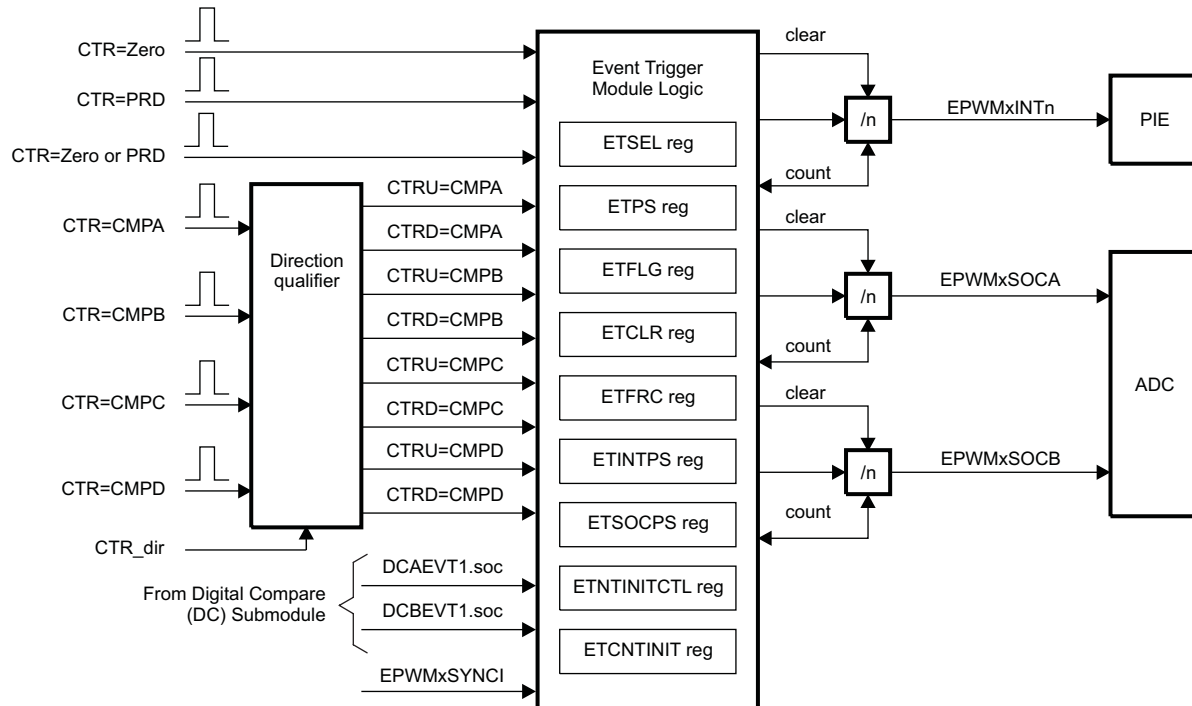
The event-trigger submodule manages the events generated by the time-base submodule, the counter-compare submodule, and the digital-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. Figure 13-43 illustrates where the event-trigger submodule fits within the ePWM system.

Figure 13-43. Event-Trigger Submodule


13.3.8.1 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 13-44](#)) 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 13-44. 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 13-45](#), [Figure 13-46](#), and [Figure 13-47](#).

[Figure 13-45](#) 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[INTPSSEL] 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 ETFLG[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 13-45. Event-Trigger Interrupt Generator

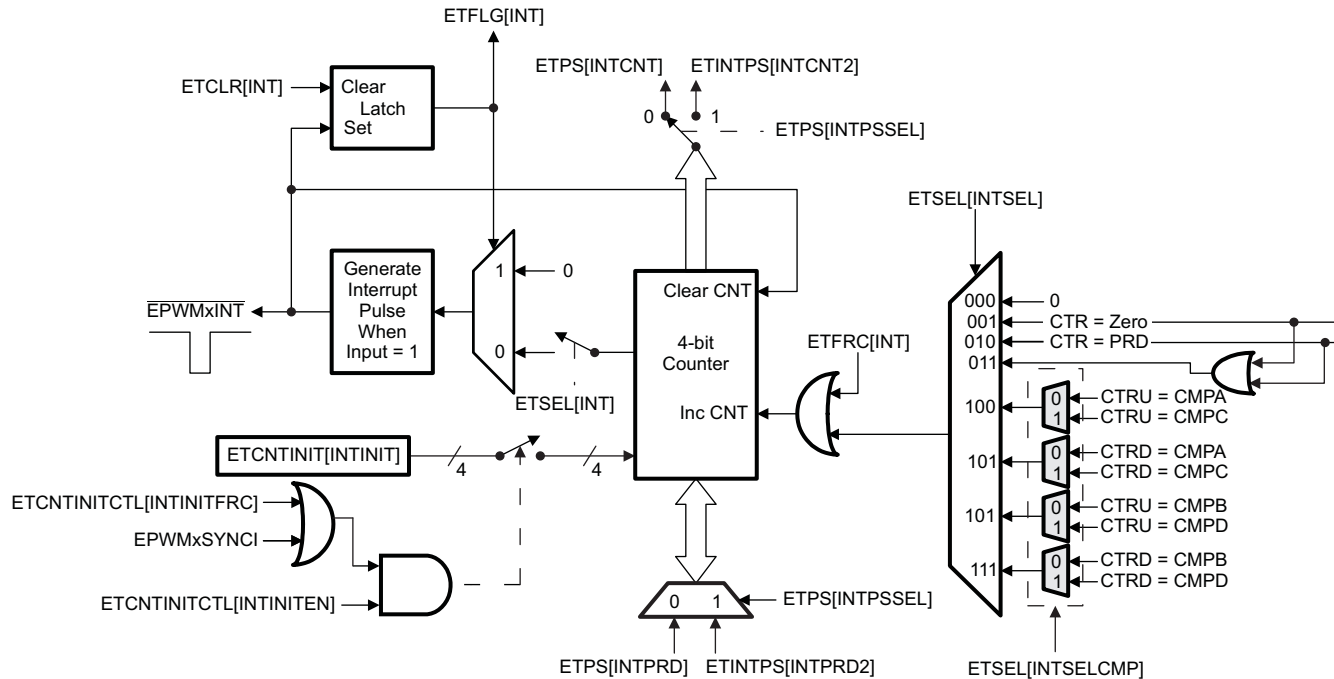


Figure 13-46 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The enhancements include SOCASELCMP and SOCBSELCMP bit fields defined in the ETSEL register enable CMPC and CMPD events respectively to cause a start of conversion. The ETPS[SOCPSSEL] bit field determines 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.

[illegible]

Figure 13-47 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.

The diagram shows the ETM SOC block with its internal components and signal flow. The 4-bit Counter has a Clear Latch Set and a Generate SOC Pulse When Input = 1 block. The ETSEL[SOCBSEL] multiplexer selects between different control signals based on the ETSEL[SOCBSEL] input. The truth table for the ETSEL[SOCBSEL] multiplexer is as follows:

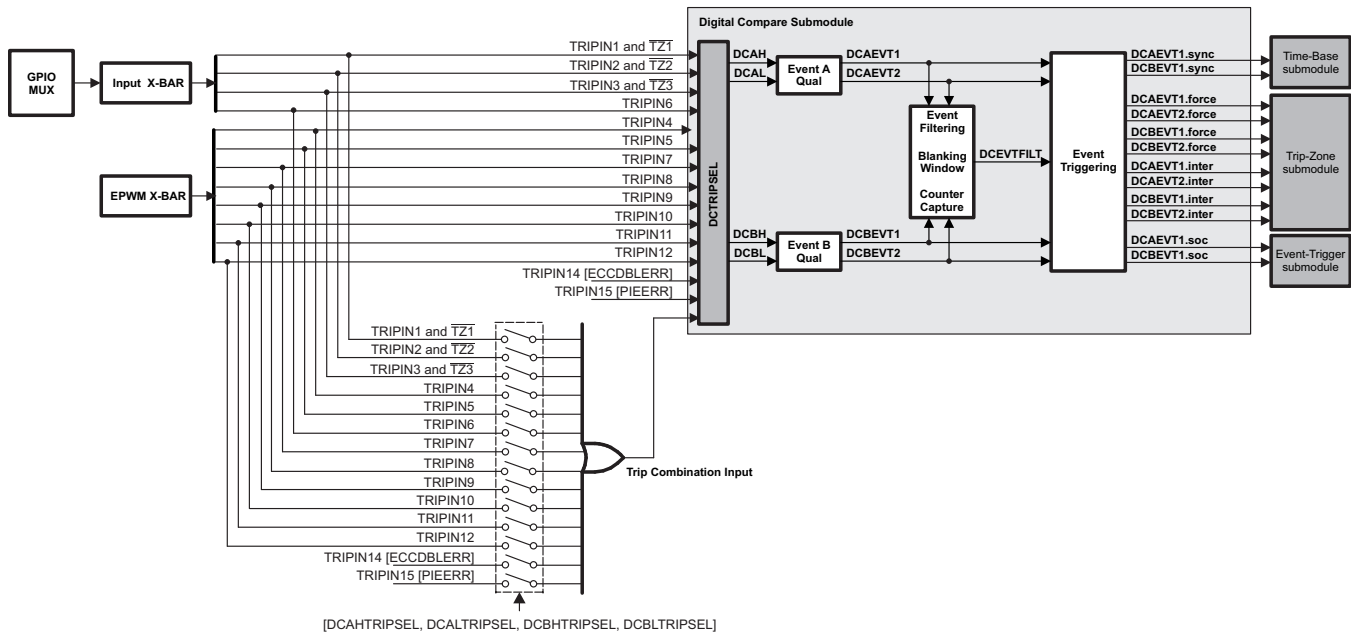
ETSEL[SOCBSEL]	Control Signal
000	DCBEVT1.soc
001	CTR = Zero
010	CTR = PRD
011	CTR = PRD
100	CTRU = CMPA, CTRU = CMPC
101	CTRD = CMPA, CTRD = CMPC
110	CTRU = CMPB, CTRU = CMPD
111	CTRD = CMPB, CTRD = CMPD

Copyright © 2013–2015, Texas Instruments Incorporated

13.3.9 Digital Compare (DC) Submodule

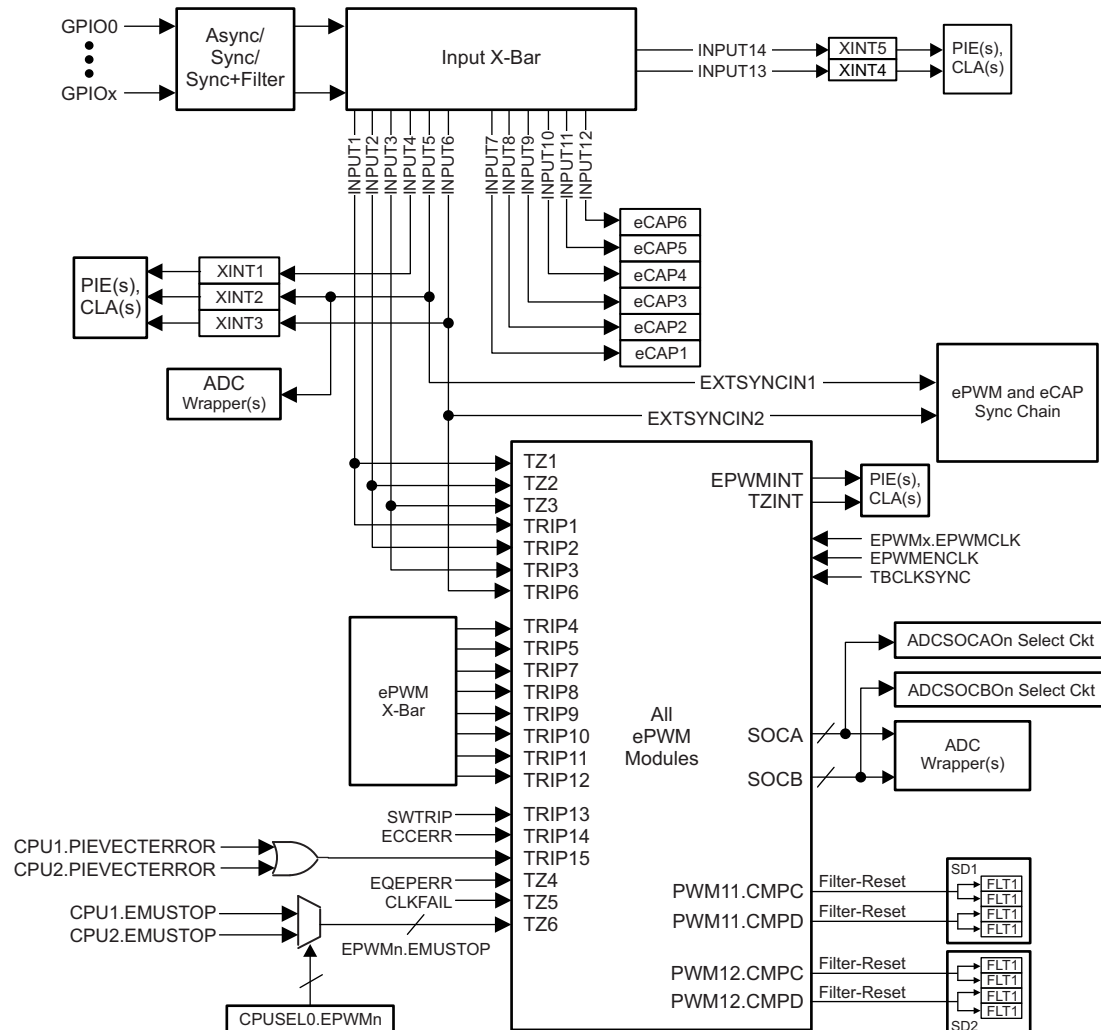
Figure 13-48 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

Figure 13-48. Digital-Compare Submodule High-Level Block Diagram



The eCAP input signals are sourced from the Input X-BAR signals as shown in Figure 13-49.

Figure 13-49. 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.

13.3.9.1 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.

13.3.9.2 Enhanced Trip Action

In order to allow multiple comparators 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 “combinational input” using the DCTRIPSEL register. This is configured by writing appropriate value [Trip Combination input] to the DCAHCOMPSEL, DCALCOMPSEL, DCBHCOMPSEL, DCBLCOMPSEL bit fields in the DCTRIPSEL register.

The user has an discrete choice for which trip input to put through the combinational logic for Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals generation. This is achieved using the selection from DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL and DCBLTRIPSEL register. The appropriate bit when set indicates that Trip input is chosen for “combinational input” by the DCTRIPSEL register.

Apart from these options user can also make the external trip inputs which feed into the OR gate individually selectable and not go through the “combinational input” by using the DCTRIPSEL register.

13.3.9.3 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 PWMSYNC 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 COMPSTCLR) or by generating an early PWMSYNC signal. The ePWM modules on this device include the ability to generate PWMSYNC upon a CMPC or CMPD match (via HRPCTL) for arbitrary PWMSYNC placement within the PWM cycle.

13.3.9.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

13.3.9.4.1 Digital Compare Events

As illustrated in [Section 13.3.9.4.2](#) 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 13.3.9.4.2](#). 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.

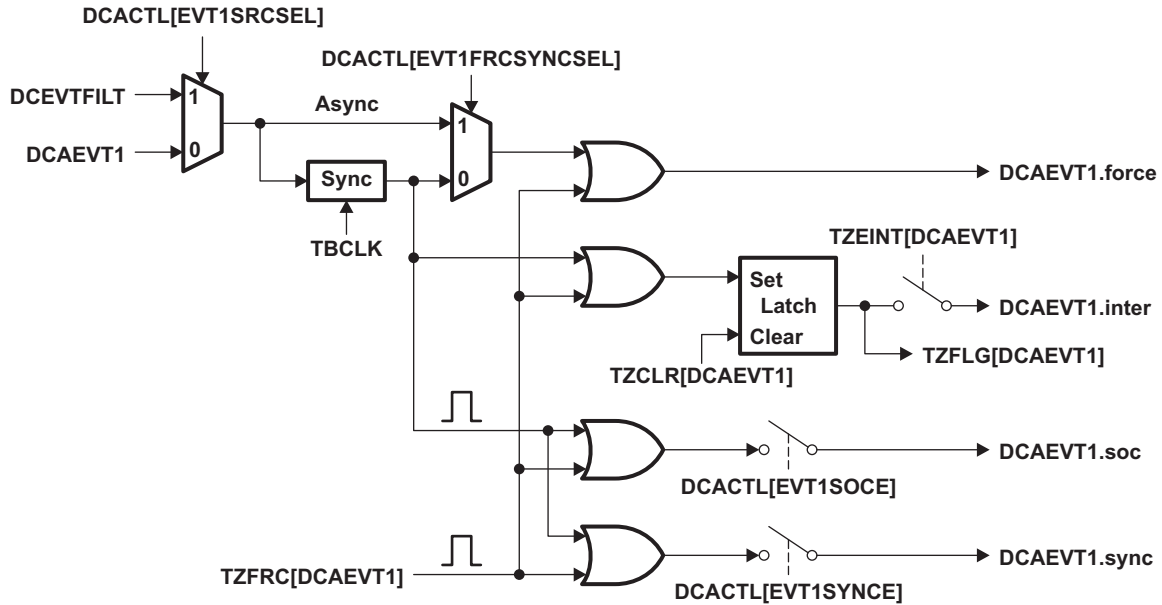
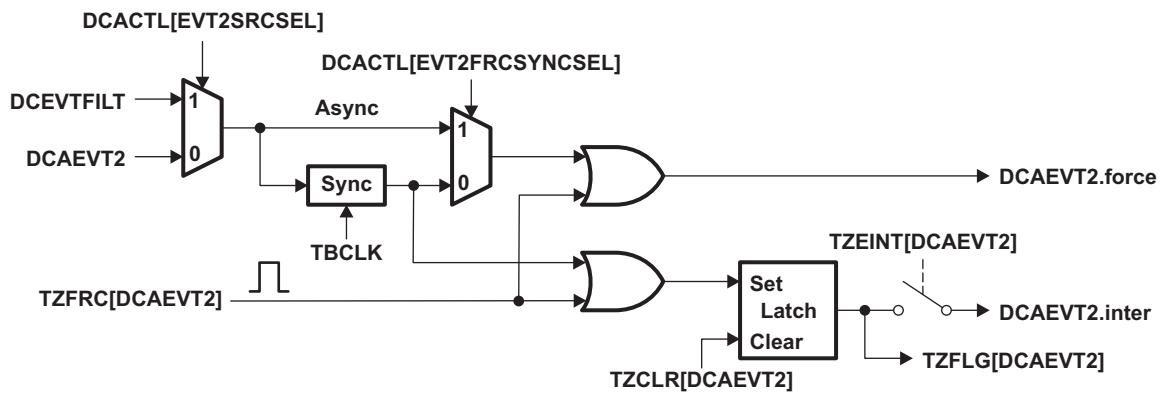
Figure 13-50. DCAEVT1 Event Triggering

Figure 13-51. DCAEVT2 Event Triggering


Figure 13-52 and Figure 13-53 show how the DCBEVT1, DCBEVT2 or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc and sync signals.

Figure 13-52. DCBEVT1 Event Triggering

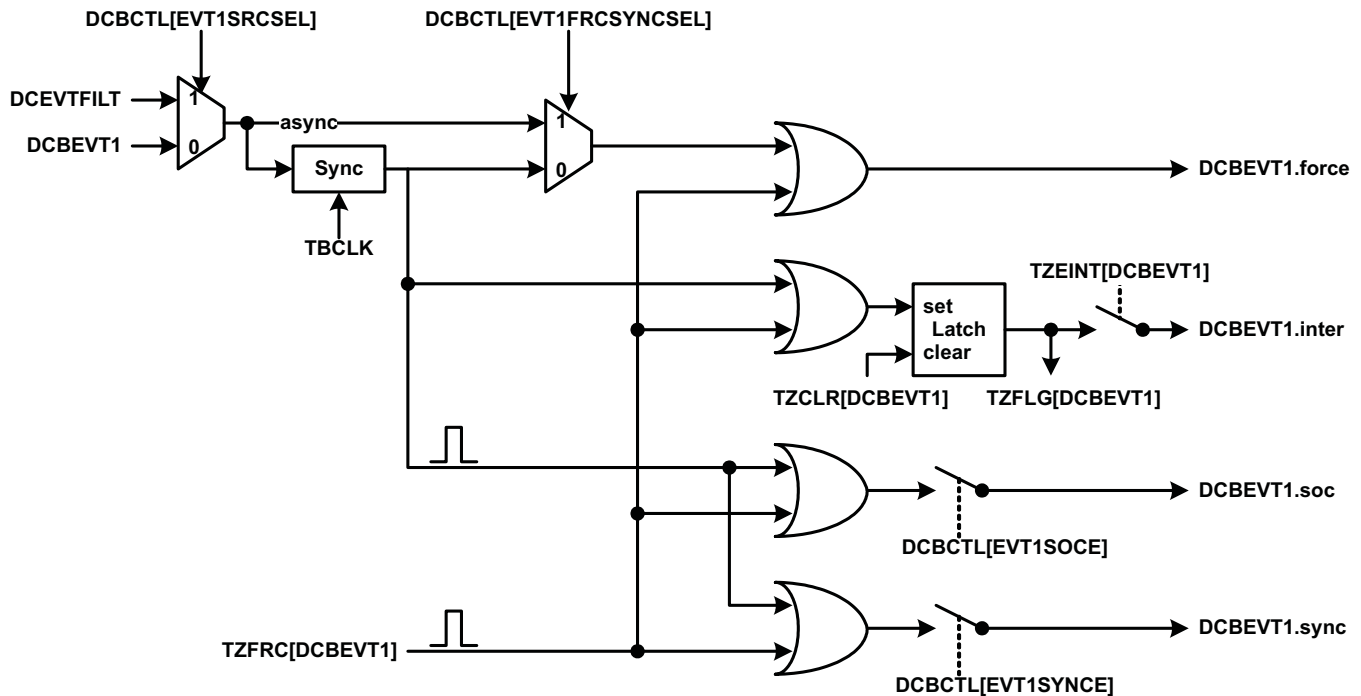
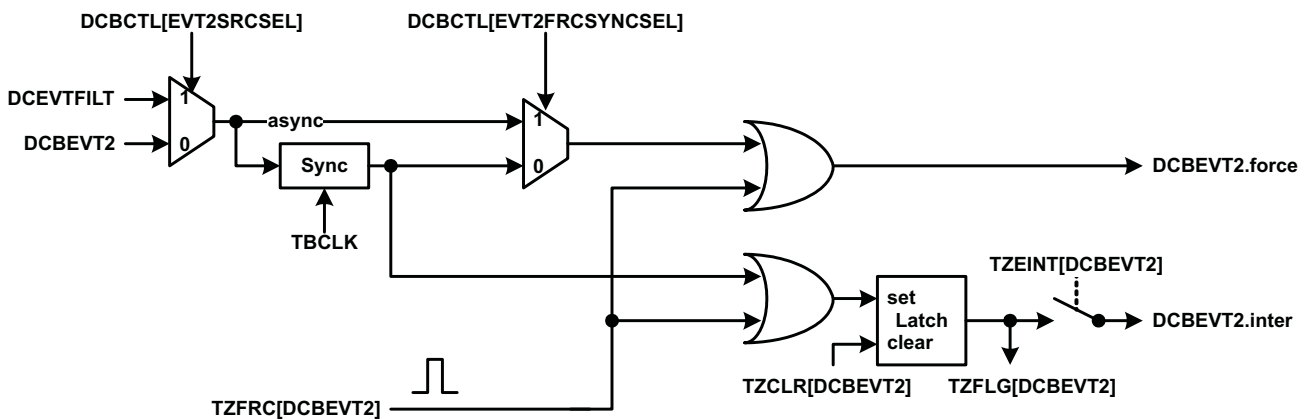
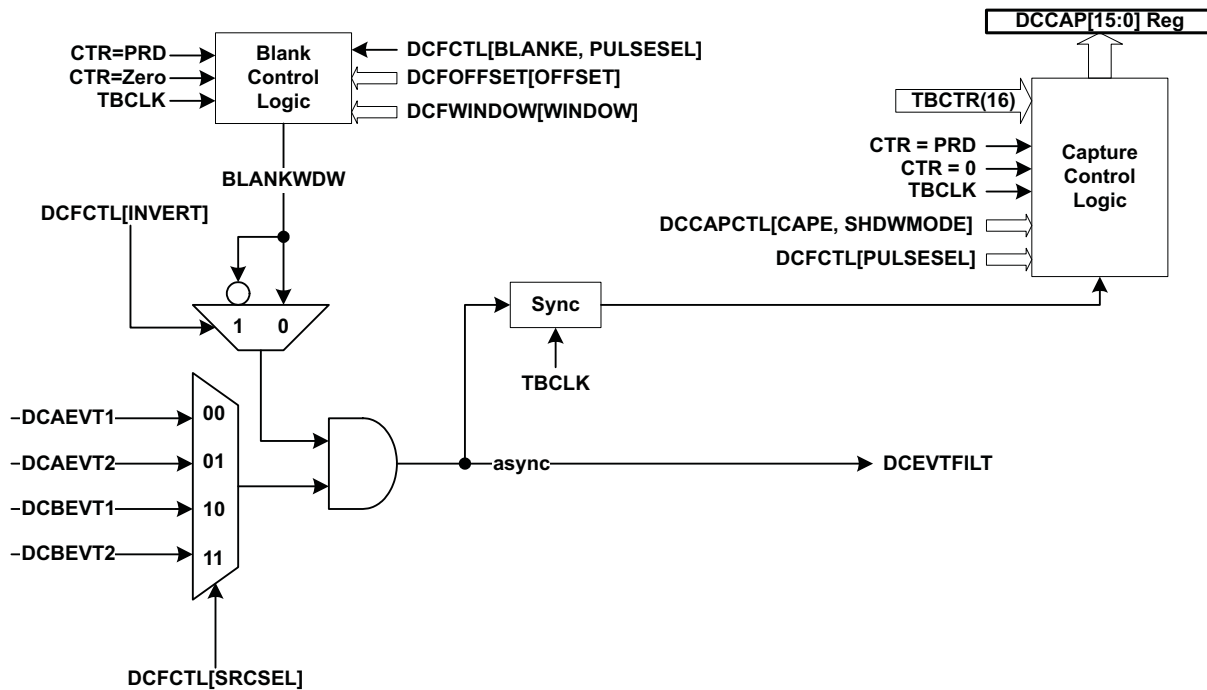


Figure 13-53. DCBEVT2 Event Triggering



13.3.9.4.2 Event Filtering

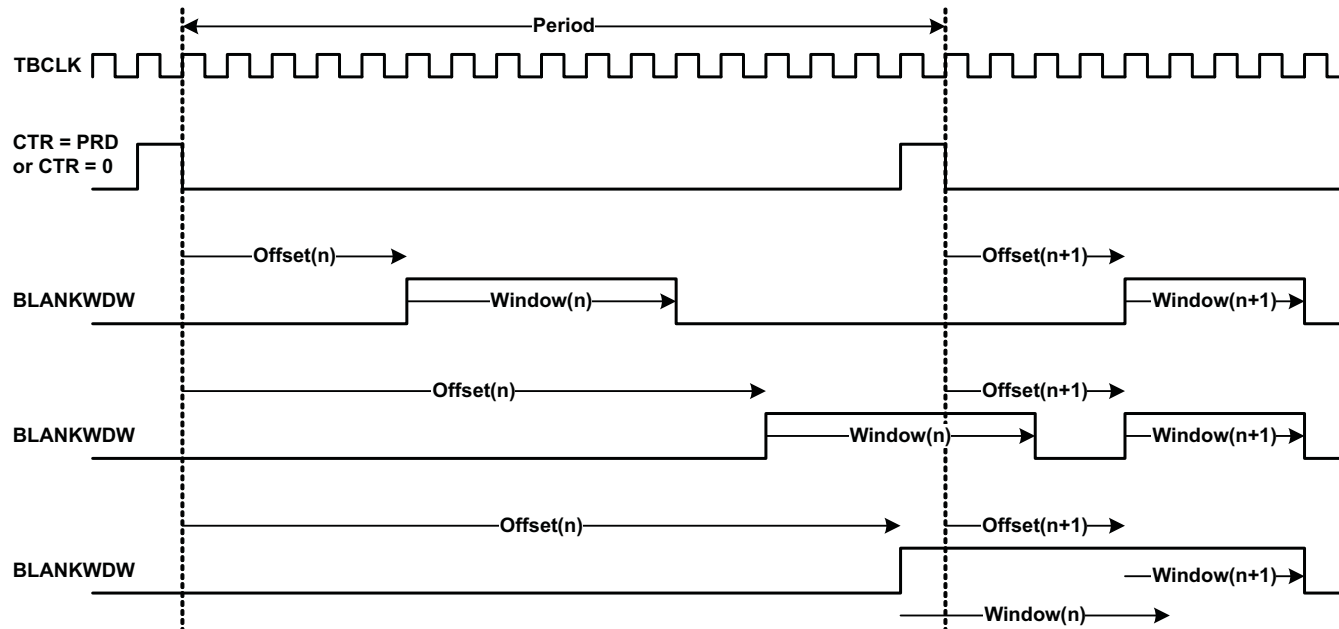
The DCAEVT1/2 and DCBEVT1/2 events can be filtered via event filtering logic to remove noise by optionally blanking events for a certain period of time. This is useful for cases where the analog comparator outputs may be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blanking logic is used 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. Figure 13-54 shows the details of the event filtering logic.

Figure 13-54. 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.

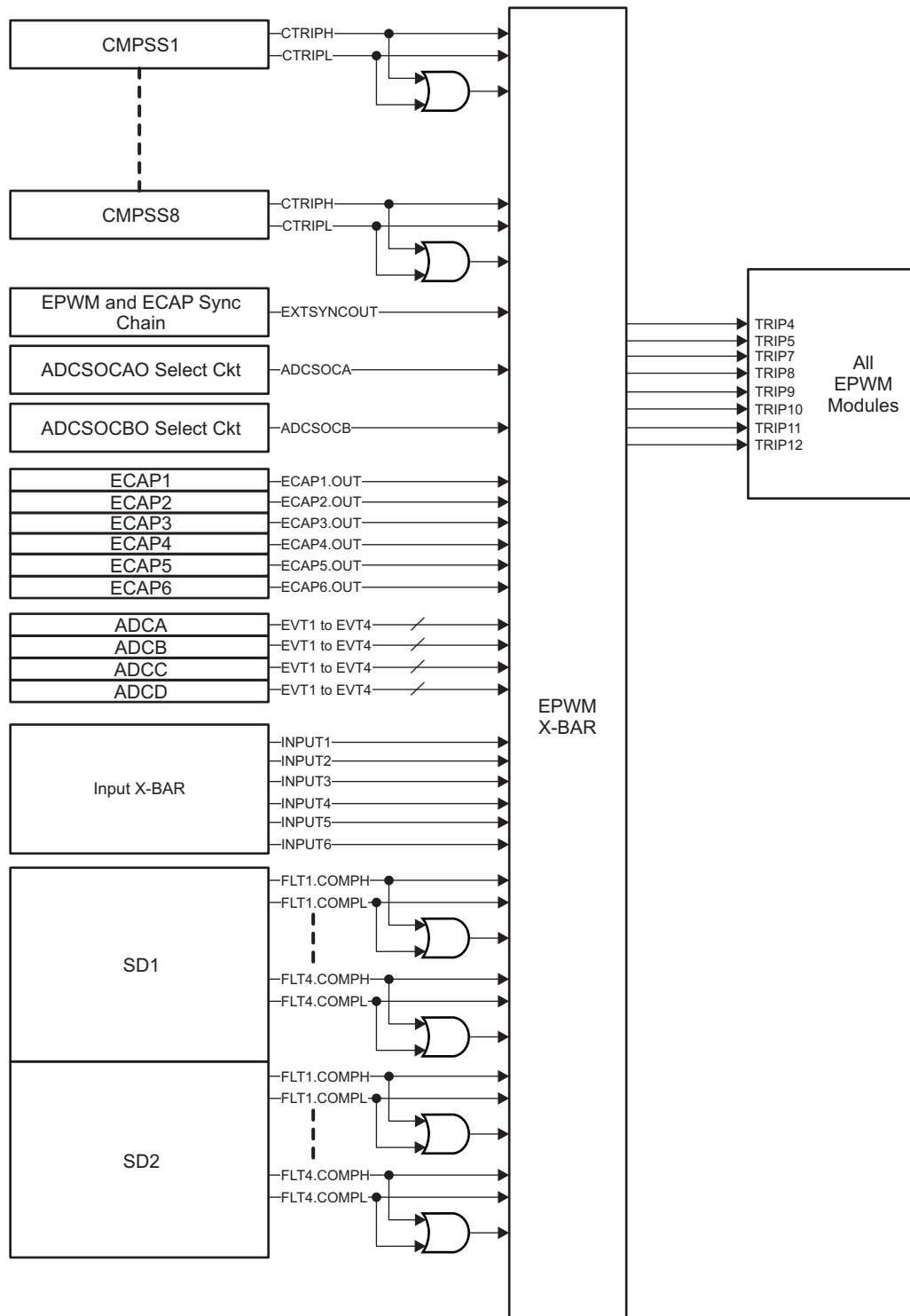
The diagram below illustrates several timing conditions for the offset and blanking window within an ePWM period. Notice that if the blanking window crosses the CTR = 0 or CTR = PRD boundary, the next window still starts at the same offset value after the CTR = 0 or CTR = PRD pulse.

Figure 13-55. Blanking Window Timing Diagram



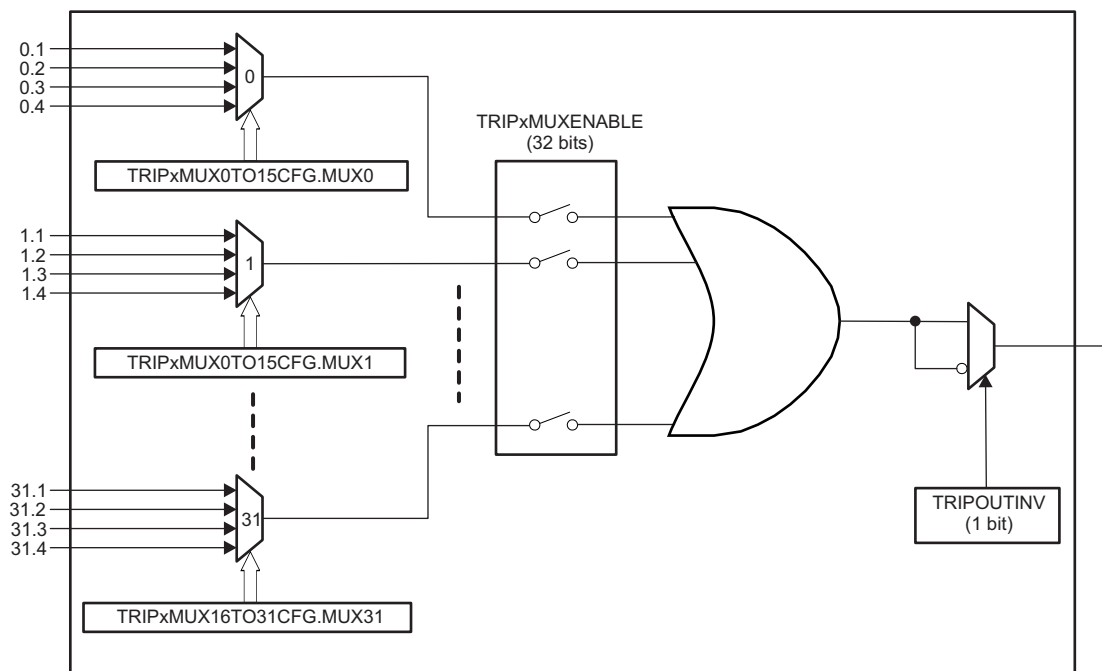
13.3.10 EPWM X-BAR

Figure 13-56 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.

Figure 13-56. EPWM X-BAR


The EPWM X-BAR has eight outputs which are routed to each EPWM module. [Figure 13-57](#) represents the architecture of a single output but it is identical to the architecture of all of the other outputs:

Figure 13-57. EPWM 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 TRIPOUTINV register.

Table 13-13. ePWM X-BAR Mux Configuration Table

Mux	1	2	3	4
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	ADCAEVT1	ECAP1.OUT
1	CMPSS1.CTRIPL	INPUTXBAR1		ADCCEVT1
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	ADCAEVT2	ECAP2.OUT
3	CMPSS2.CTRIPL	INPUTXBAR2		ADCCEVT2
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	ADCAEVT3	ECAP3.OUT
5	CMPSS3.CTRIPL	INPUTXBAR3		ADCCEVT3
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	ADCAEVT4	ECAP4.OUT
7	CMPSS4.CTRIPL	INPUTXBAR4		ADCCEVT4
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	ADCBEVT1	ECAP5.OUT
9	CMPSS5.CTRIPL	INPUTXBAR5		ADCDEVT1
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	ADCBEVT2	ECAP6.OUT
11	CMPSS6.CTRIPL	INPUTXBAR6		ADCDEVT2
12	CMPSS7.CTRIPH	CMPSS7.CTRIPH_OR_CTRIPL	ADCBEVT3	
13	CMPSS7.CTRIPL	ADCSOCA		ADCDEVT3
14	CMPSS8.CTRIPH	CMPSS8.CTRIPH_OR_CTRIPL	ADCBEVT4	EXTSYNCOUT
15	CMPSS8.CTRIPL	ADCSOCB		ADCDEVT4
16	SD1FLT1.COMPH	SD1FLT1.COMPH_OR_COMPL		
17	SD1FLT1.COMPL			
18	SD1FLT2.COMPH	SD1FLT2.COMPH_OR_COMPL		
19	SD1FLT2.COMPL			
20	SD1FLT3.COMPH	SD1FLT3.COMPH_OR_COMPL		
21	SD1FLT3.COMPL			
22	SD1FLT4.COMPH	SD1FLT4.COMPH_OR_COMPL		
23	SD1FLT4.COMPL			
24	SD2FLT1.COMPH	SD2FLT1.COMPH_OR_COMPL		
25	SD2FLT1.COMPL			
26	SD2FLT2.COMPH	SD2FLT2.COMPH_OR_COMPL		
27	SD2FLT2.COMPL			
28	SD2FLT3.COMPH	SD2FLT3.COMPH_OR_COMPL		
29	SD2FLT3.COMPL			
30	SD2FLT4.COMPH	SD2FLT4.COMPH_OR_COMPL		
31	SD2FLT4.COMPL			

Note: All unused and reserved positions are tied to '0'.

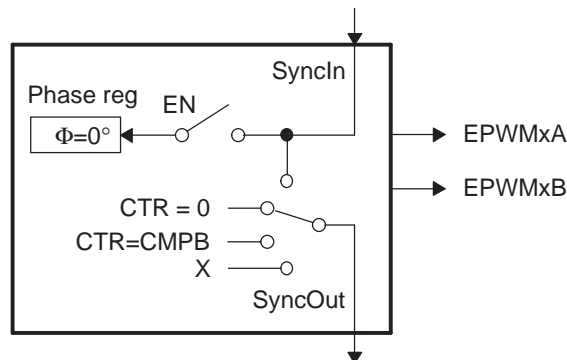
13.4 Applications to Power Topologies

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.

13.4.1 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 13-58](#). 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 13-58. Simplified ePWM Module

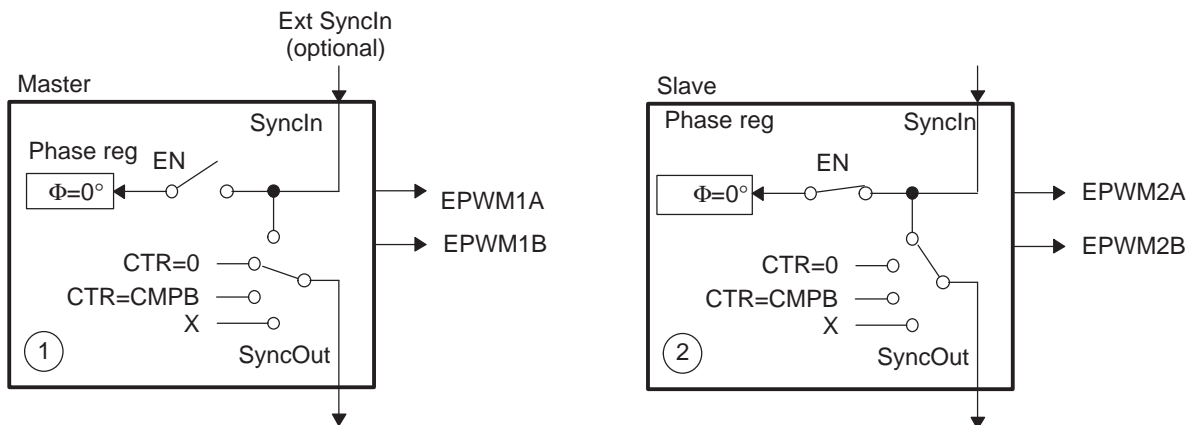


13.4.2 Key Configuration Capabilities

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)

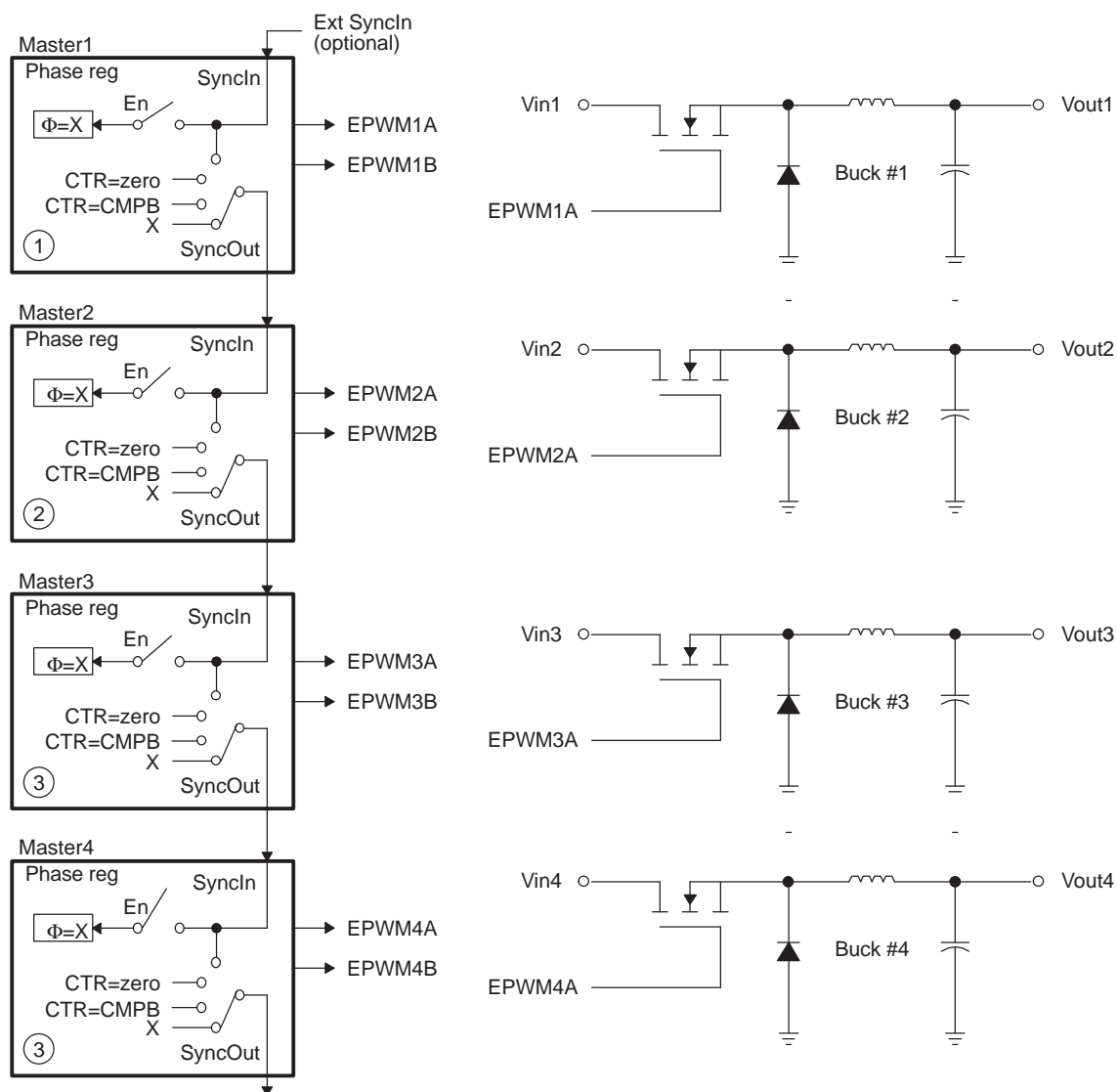
For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it (that is, via the enable switch). Although various combinations are possible, the two most common—master module and slave module modes—are shown in [Figure 13-59](#).

Figure 13-59. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave


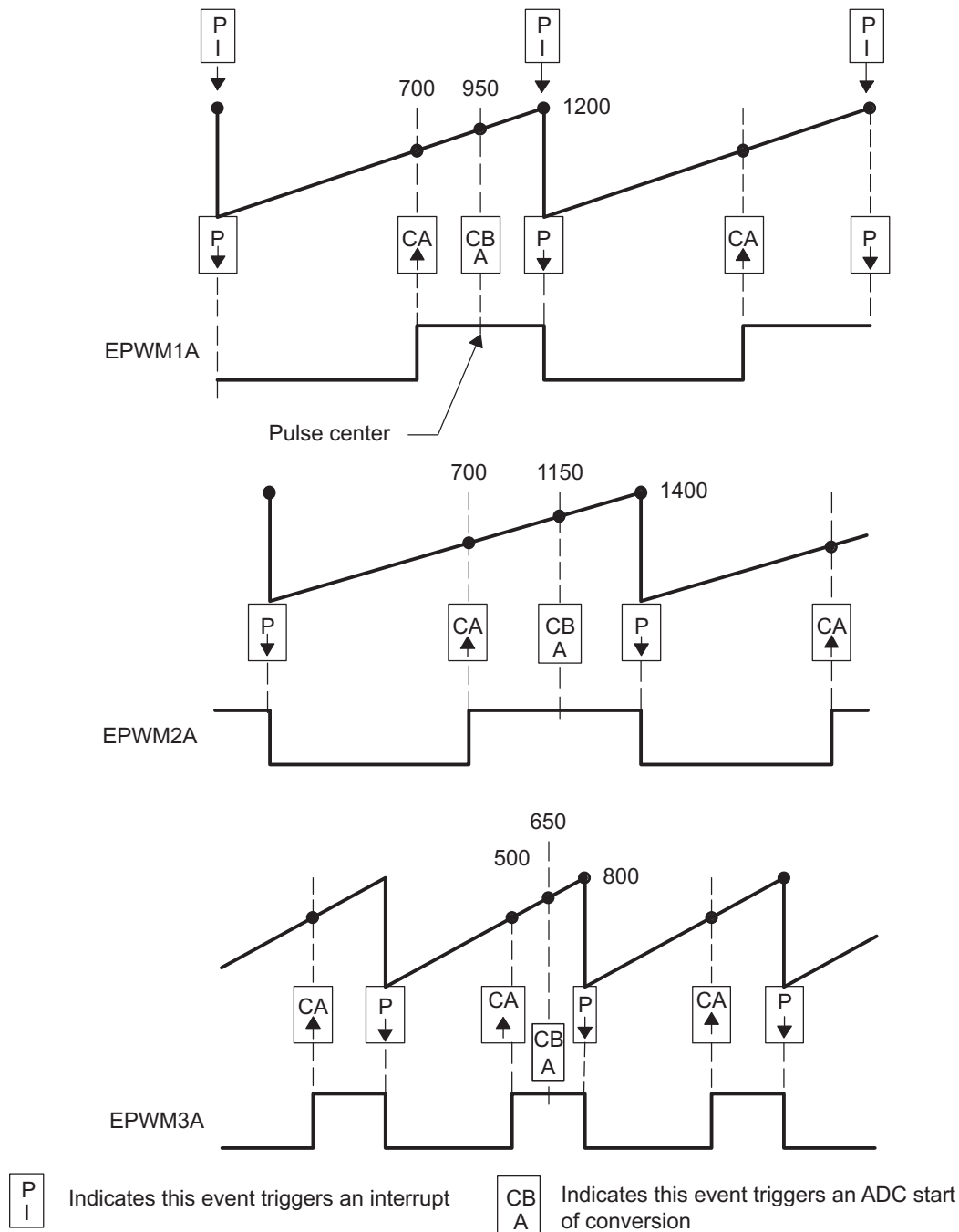
13.4.3 Controlling Multiple Buck Converters With Independent Frequencies

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 13-60](#) 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 13-61](#) shows the waveforms generated by the setup shown in [Figure 13-60](#); note that only three waveforms are shown, although there are four stages.

Figure 13-60. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$



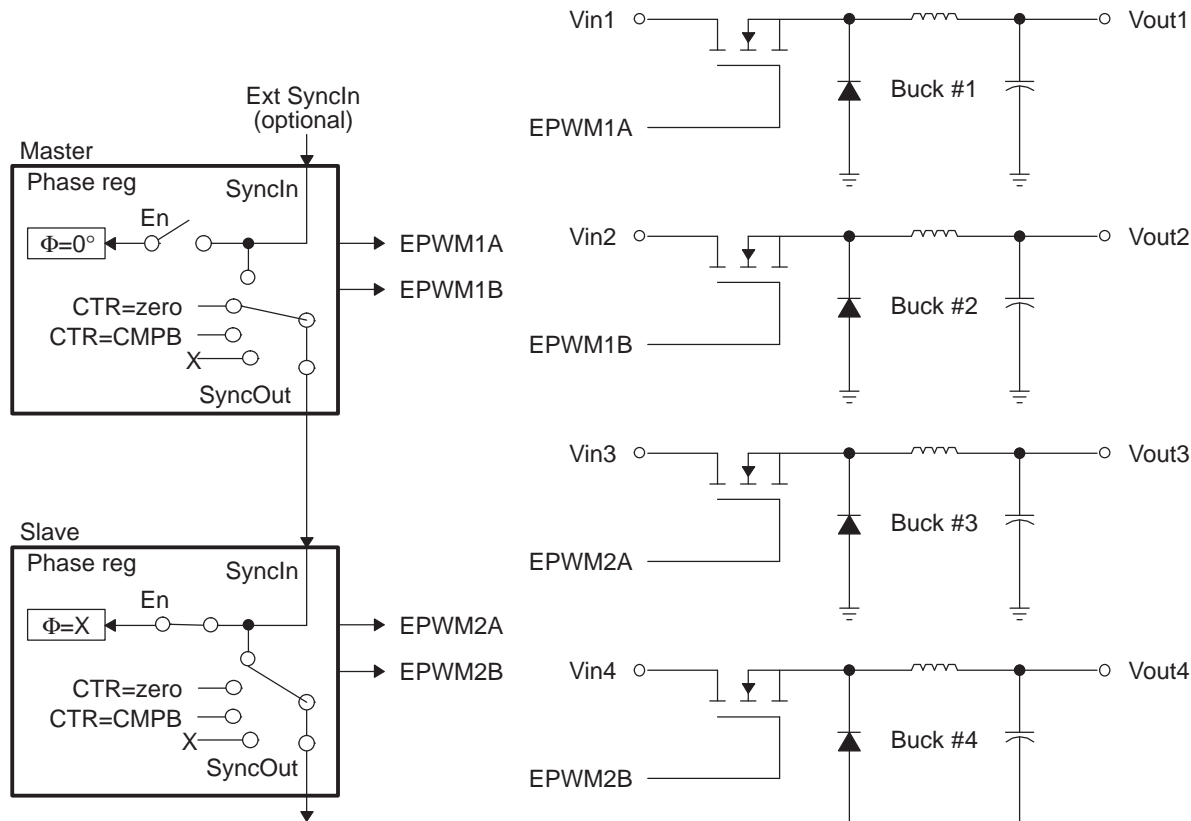
NOTE: $\phi = X$ indicates value in phase register is a "don't care"

Figure 13-61. Buck Waveforms for Figure 13-60 (Note: Only three bucks shown here)


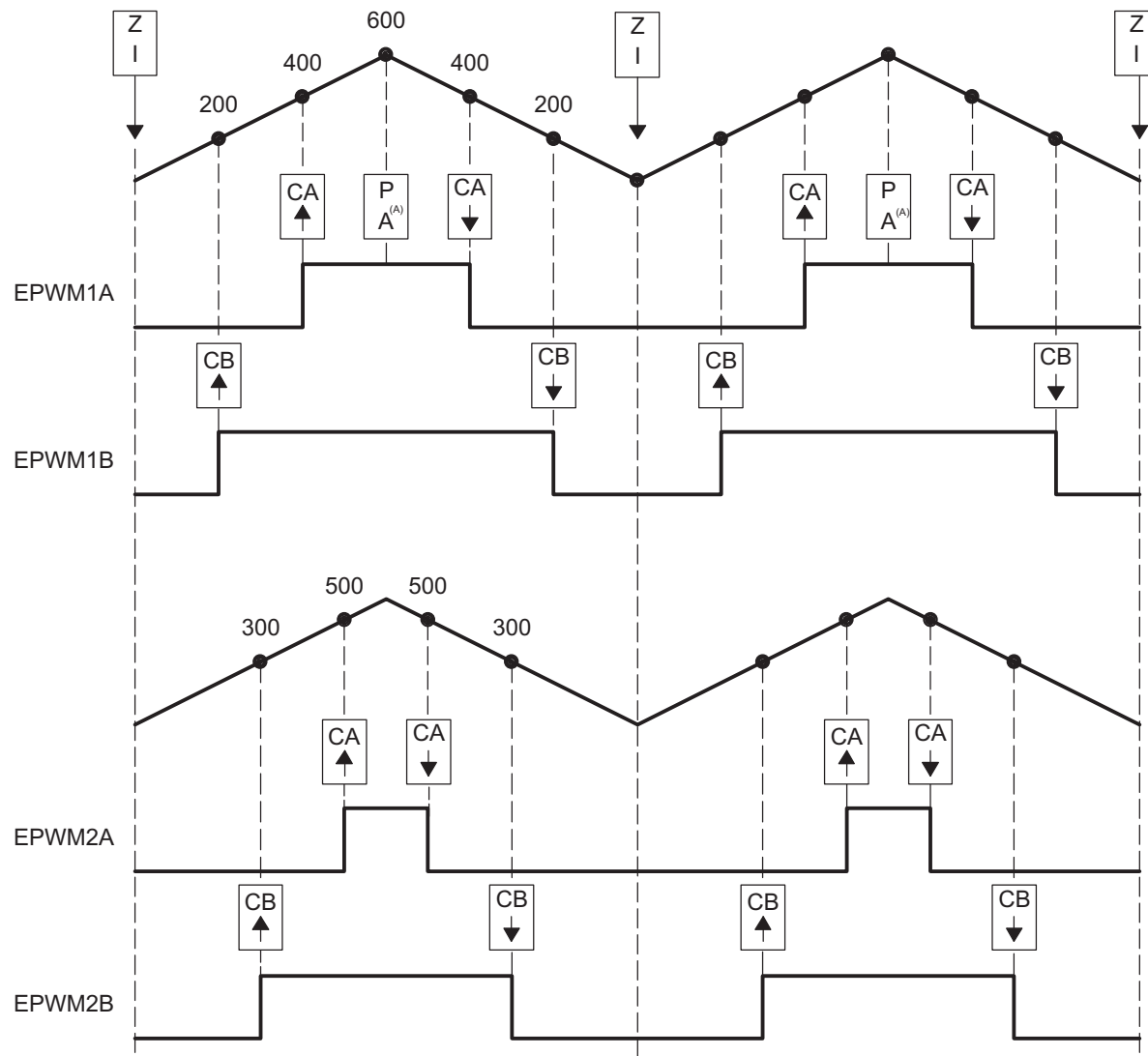
13.4.4 Controlling Multiple Buck Converters With Same Frequencies

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 13-62 shows such a configuration; Figure 13-63 shows the waveforms generated by the configuration.

Figure 13-62. 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 13-63. Buck Waveforms for Figure 13-62 (Note: $F_{PWM2} = F_{PWM1}$)


A Starts ADC conversion.

13.4.5 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 13-64 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 13-65 shows the waveforms generated by the configuration shown in Figure 13-64.

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 13-64. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$)

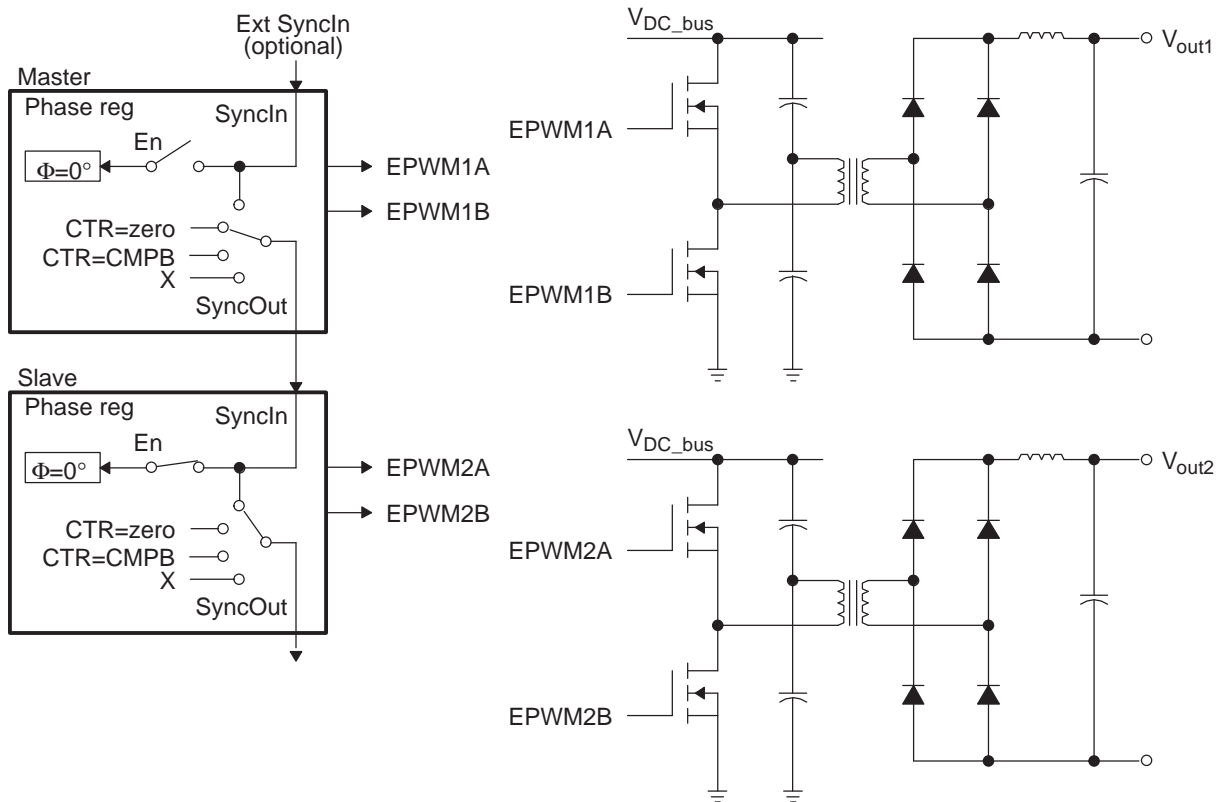
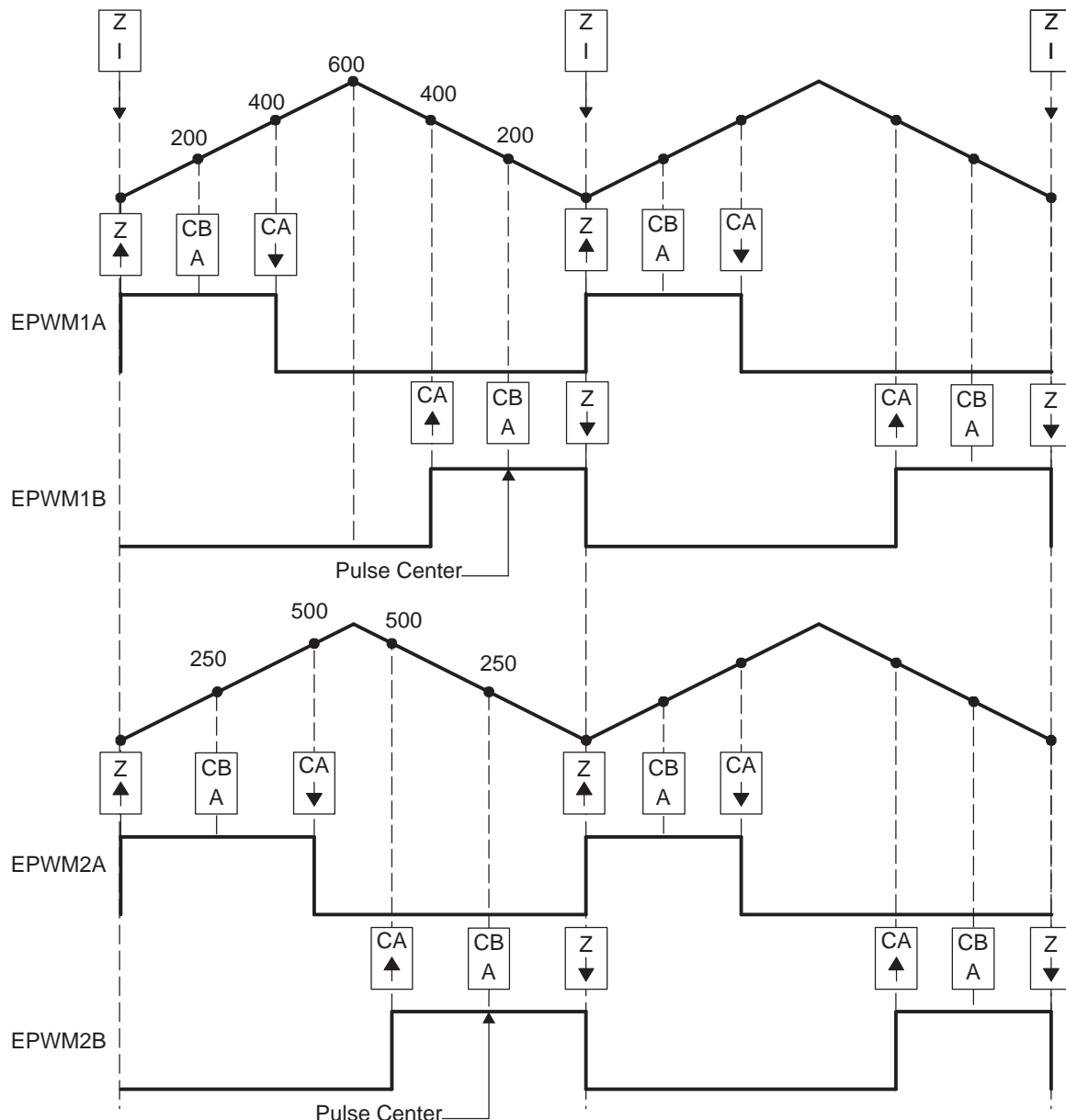


Figure 13-65. Half-H Bridge Waveforms for Figure 13-64 (Note: Here $F_{PWM2} = F_{PWM1}$)


13.4.6 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 13-66 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 13-66), 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 13-66. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control

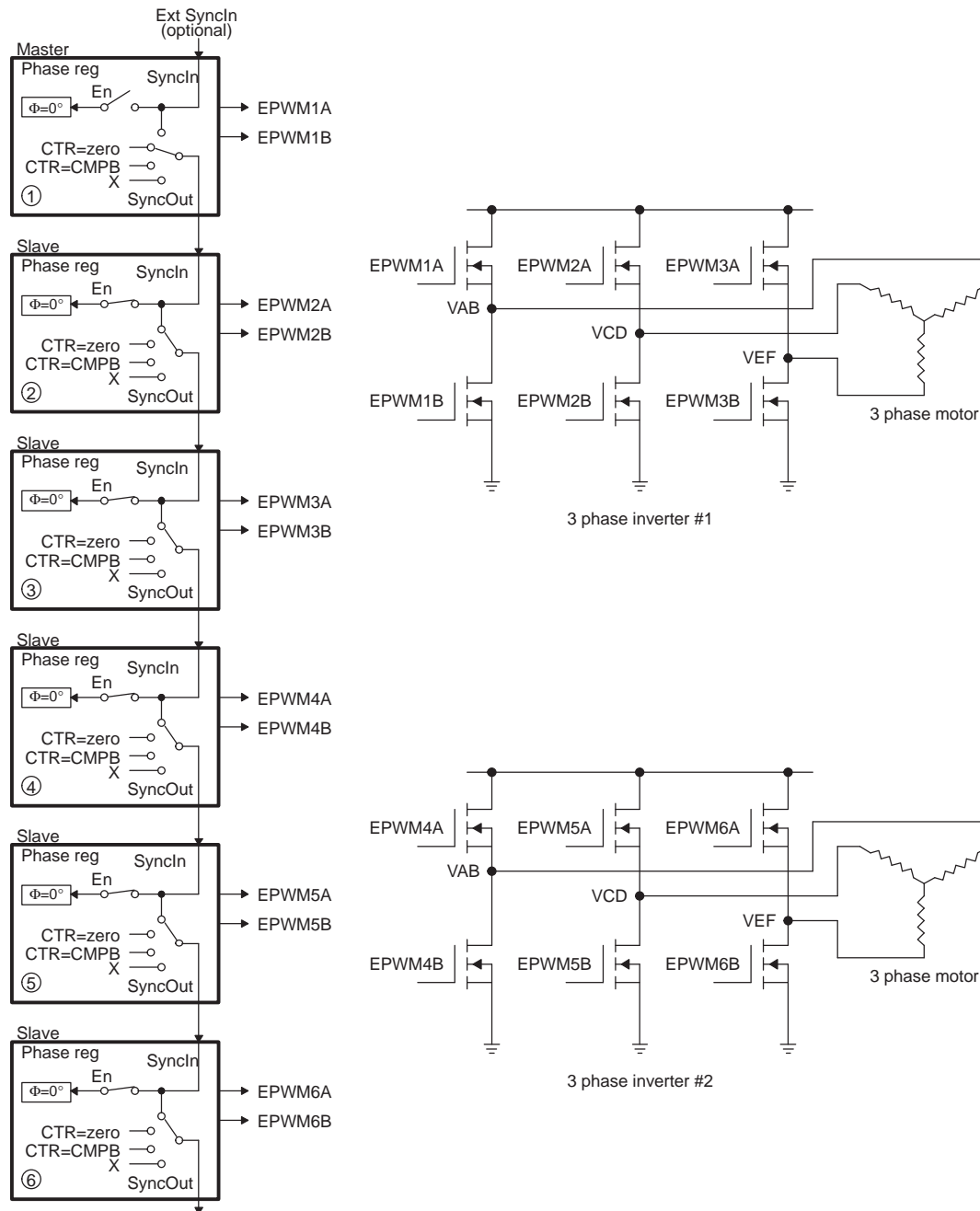


Figure 13-67. 3-Phase Inverter Waveforms for Figure 13-66 (Only One Inverter Shown)

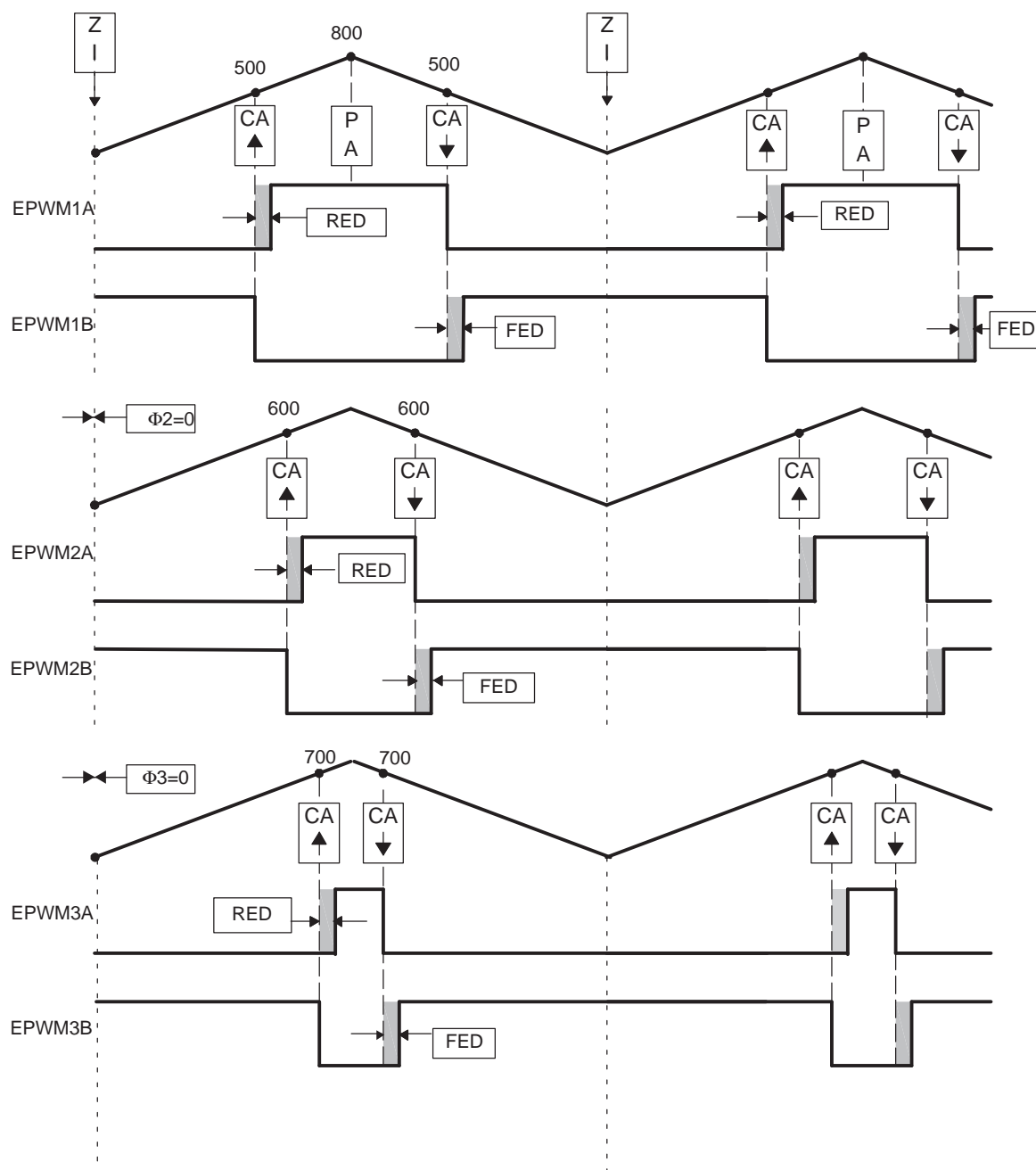
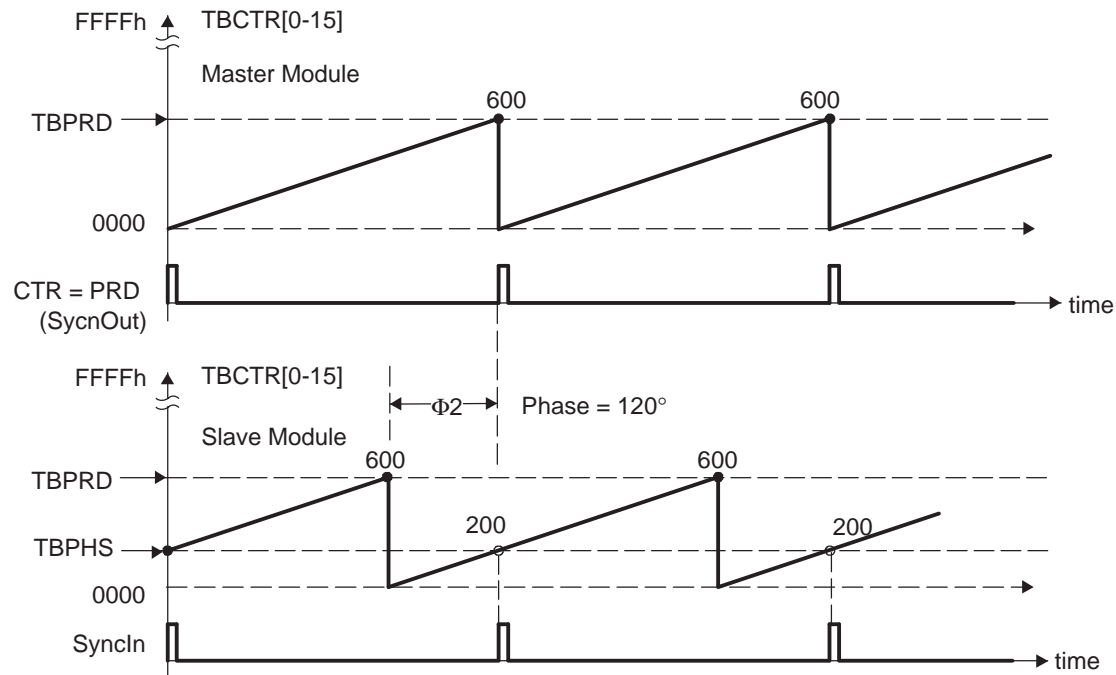


Figure 13-69. Timing Waveforms Associated With Phase Control Between Two Modules


13.4.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in [Figure 13-70](#). 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:

$$\text{TBPHS}(N,M) = (\text{TBPRD}/N) \times (M-1)$$

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N=3), TBPRD = 600,

$$\text{TBPHS}(3,2) = (600/3) \times (2-1) = 200 \text{ (that is, Phase value for Slave module 2)}$$

$$\text{TBPHS}(3,3) = 400 \text{ (that is, Phase value for Slave module 3)}$$

[Figure 13-71](#) shows the waveforms for the configuration in [Figure 13-70](#).

Figure 13-70. Control of a 3-Phase Interleaved DC/DC Converter

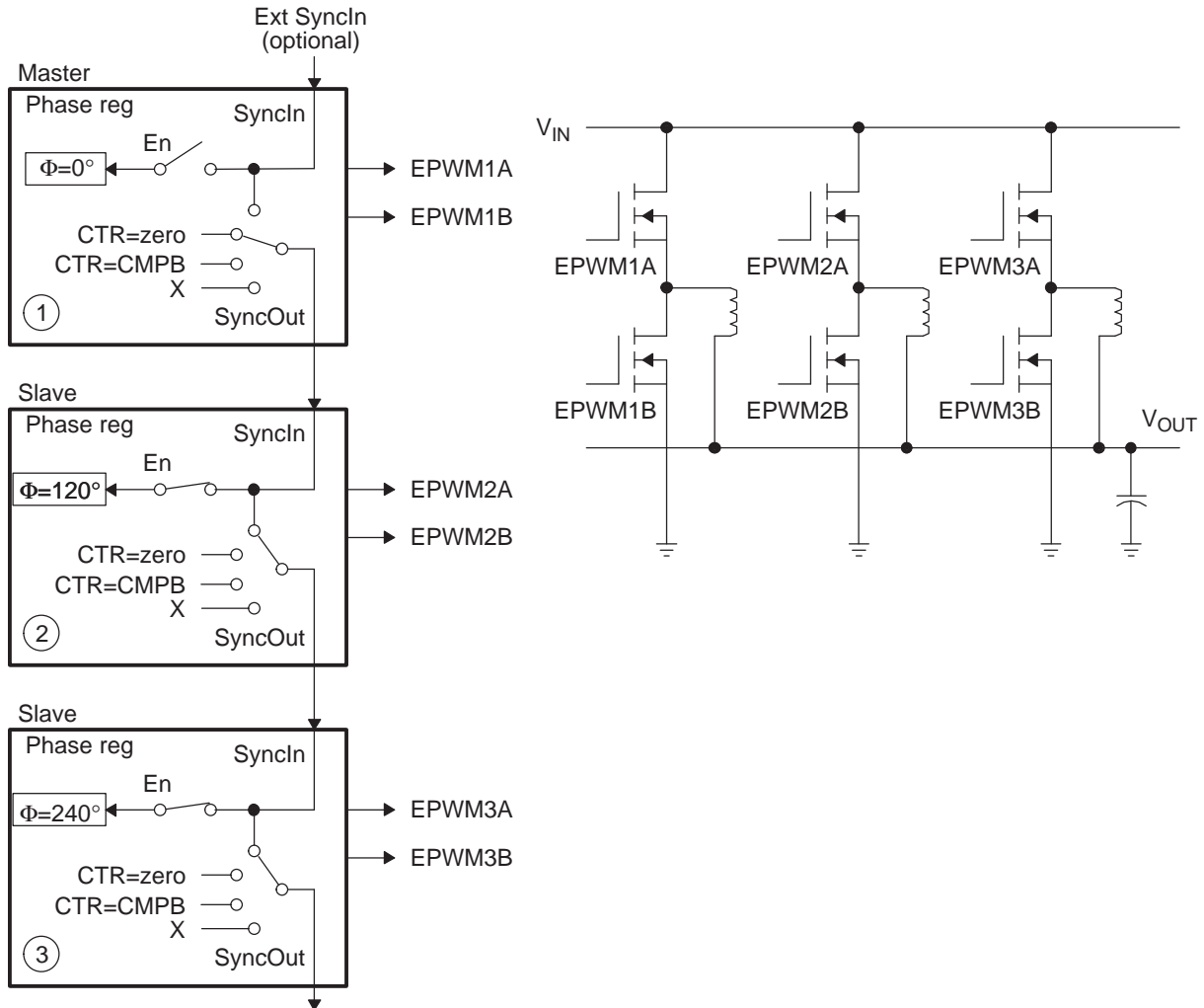
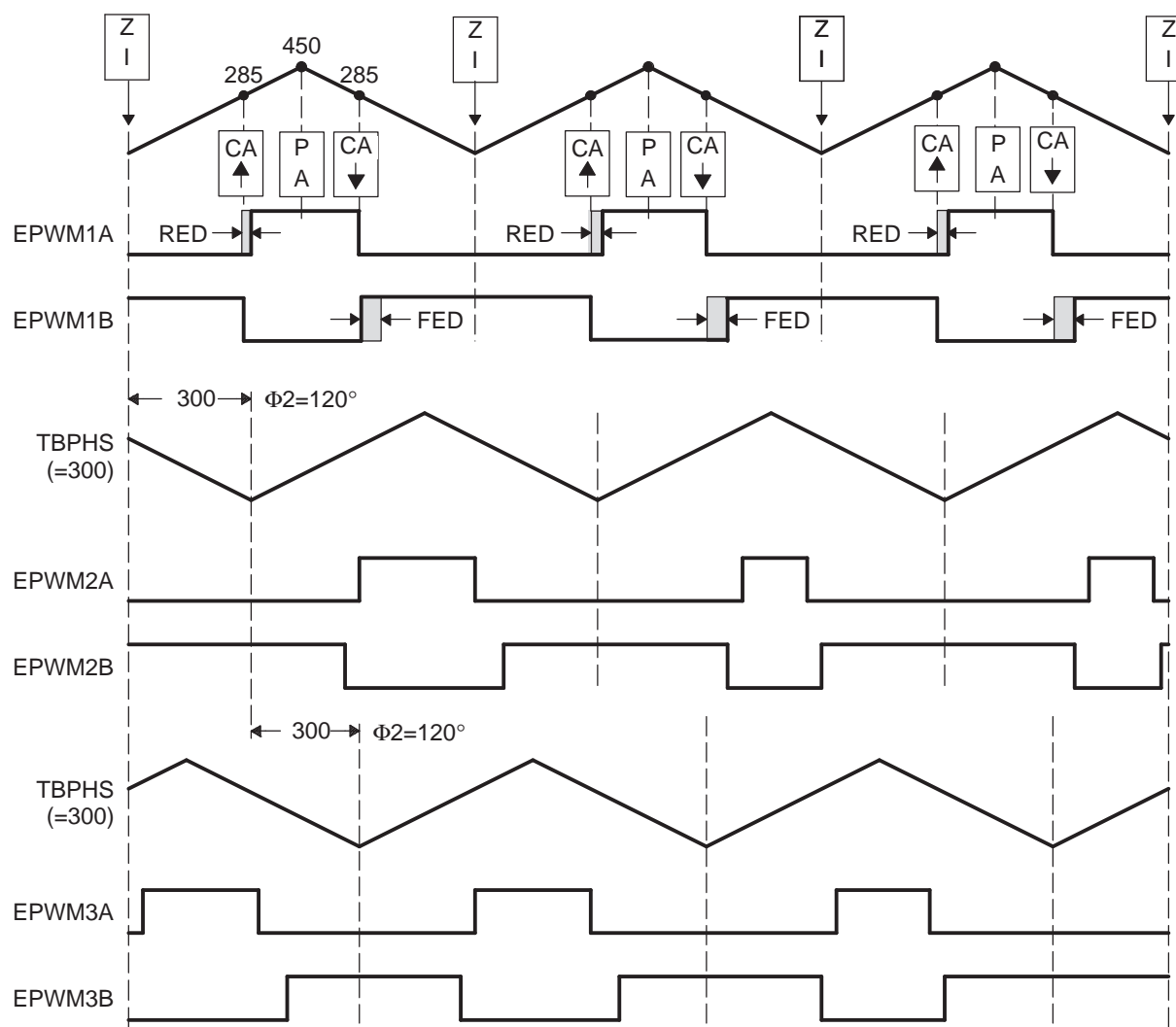


Figure 13-71. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 13-70



13.4.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 13-72 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 13-73 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 13-72. Controlling a Full-H Bridge Stage ($F_{PWM2} = F_{PWM1}$)

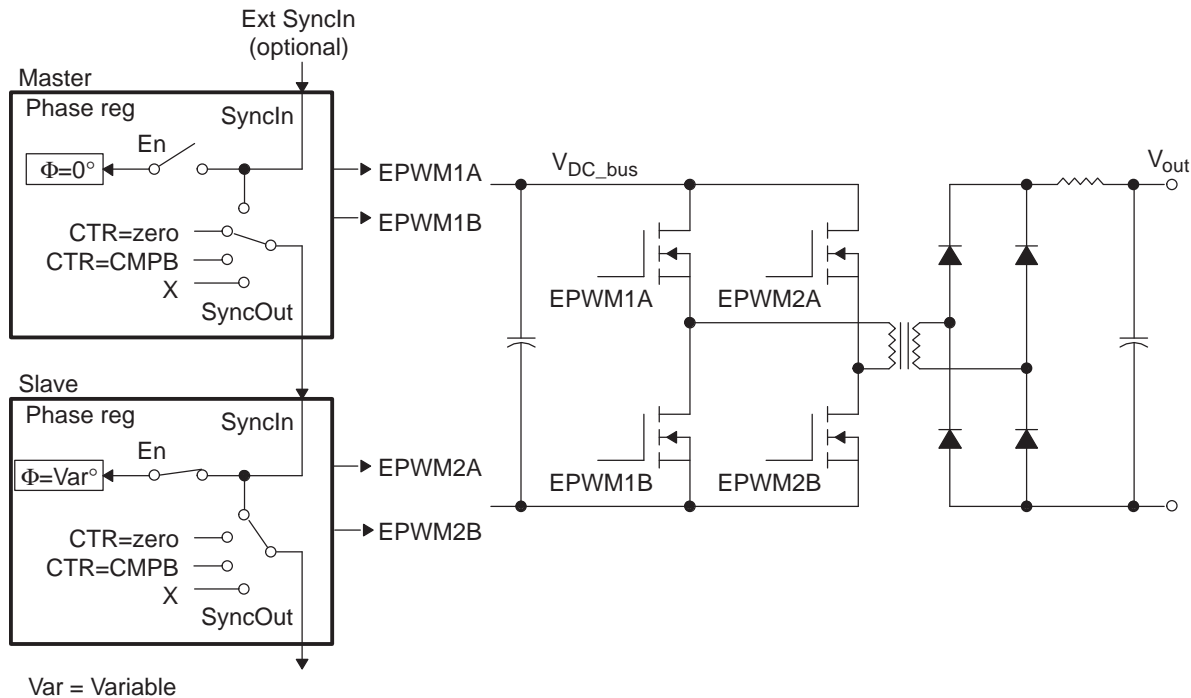
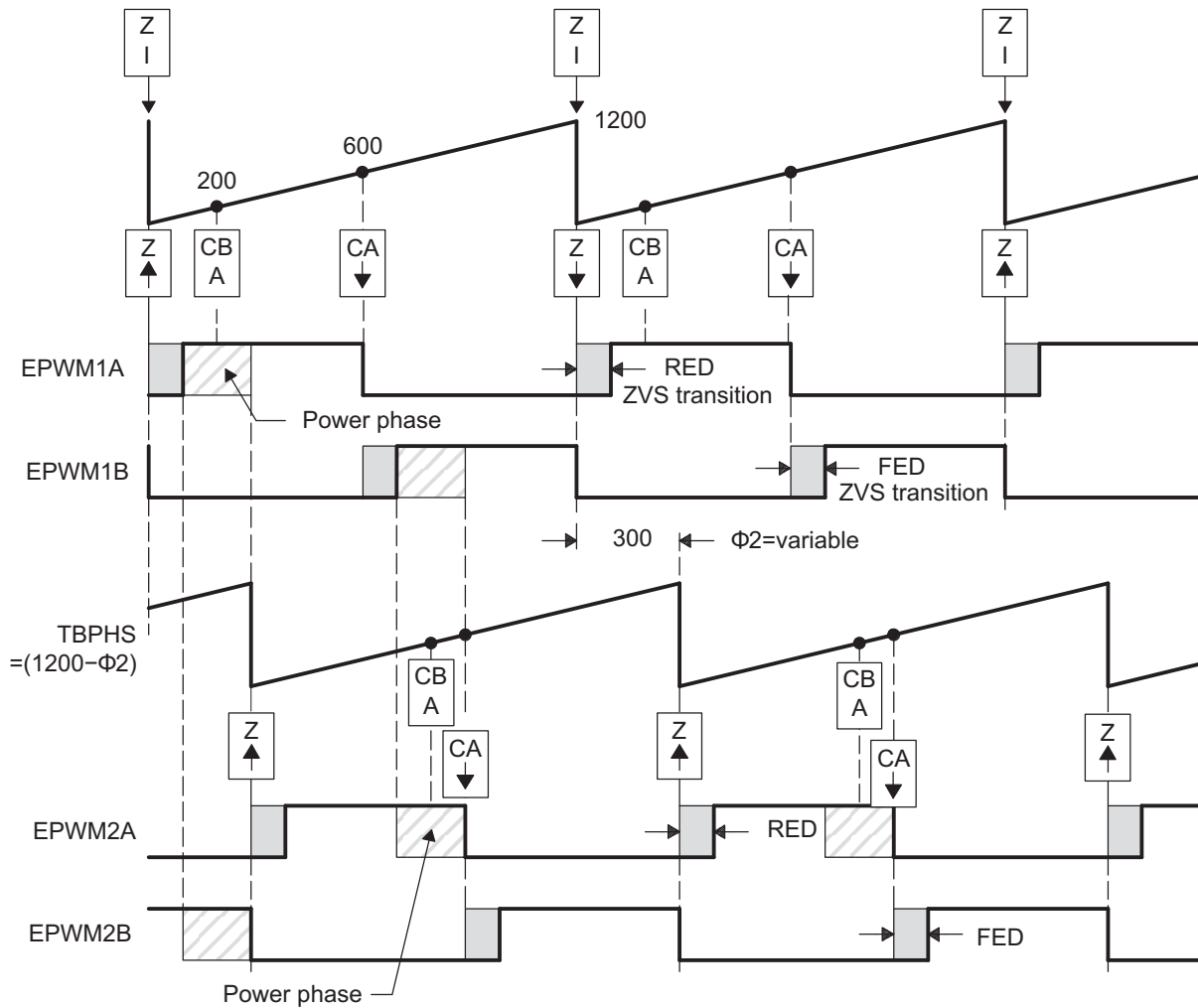


Figure 13-73. ZVS Full-H Bridge Waveforms


13.4.10 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 13-74](#) 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 13-75](#) shows the waveforms generated by the configuration.

Figure 13-74. Peak Current Mode Control of a Buck Converter

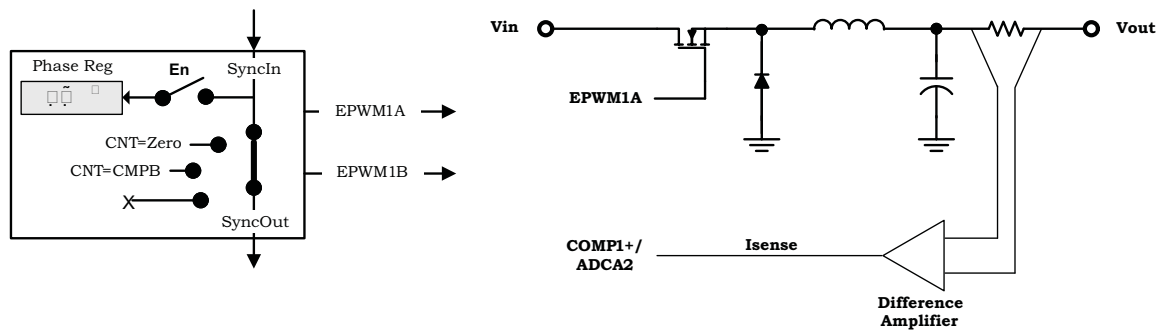
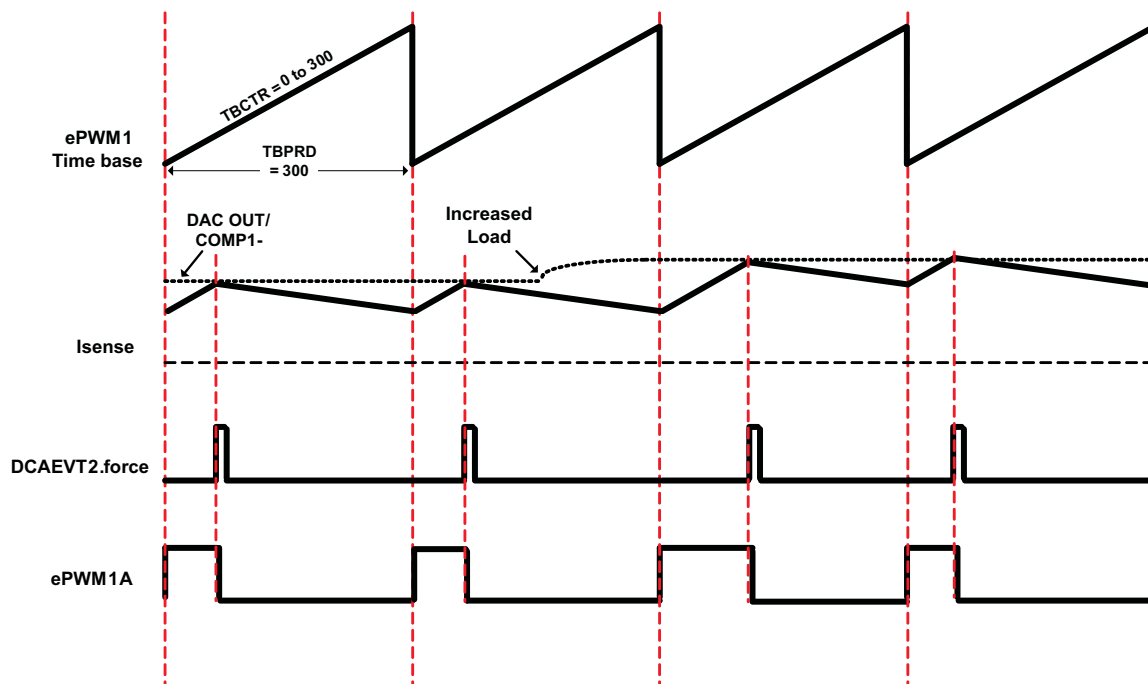


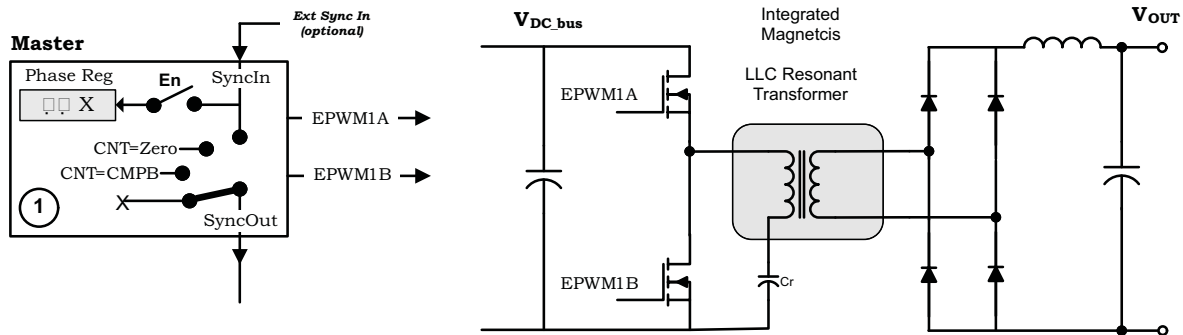
Figure 13-75. Peak Current Mode Control Waveforms for Figure 13-74



13.4.11 Controlling H-Bridge LLC Resonant Converter

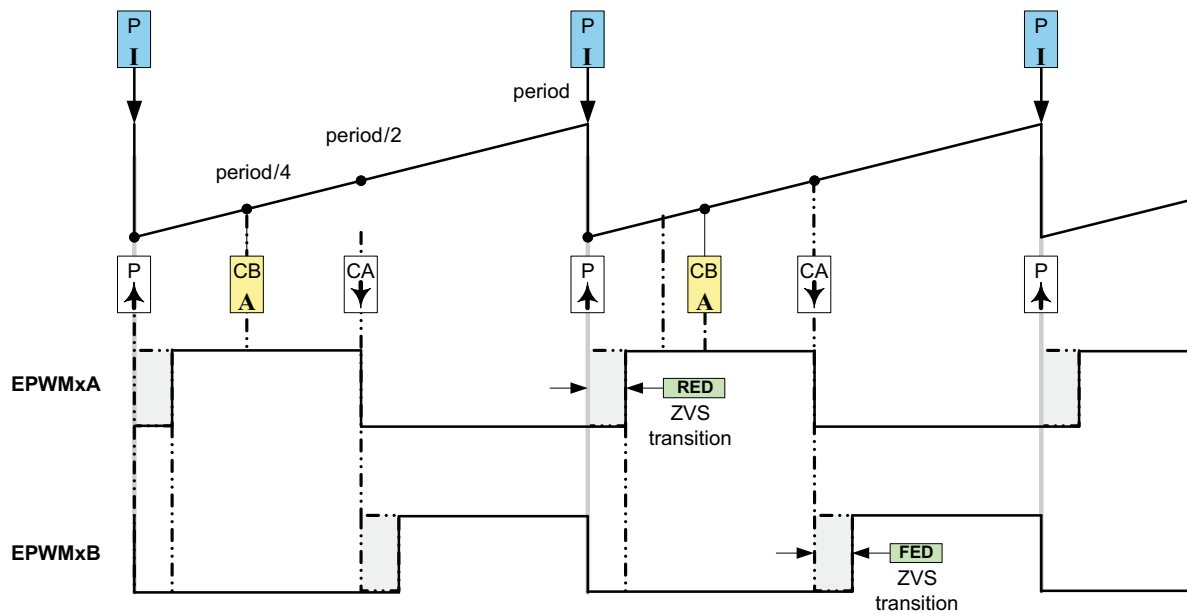
Various topologies of resonant converters are well-known in the field of power electronics for many years. In addition to these, H-bridge LLC resonant converter topology has recently gained popularity in many consumer electronics applications where high efficiency and power density are required. In this 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 13-76. Control of Two Resonant Converter Stages



NOTE: $\Theta = X$ indicates value in phase register is "don't care"

Figure 13-77. H-Bridge LLC Resonant Converter PWM Waveforms



Indicates this event triggers an interrupt



Indicates this event triggers an ADC start of conversion

13.5 Registers

13.5.1 EPWM Base Addresses

Table 13-14. EPWM Base Address Table

Device Register	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
EPwm9Regs	EPWM_REGS	0x0000_4800	0x0000_48FF
EPwm10Regs	EPWM_REGS	0x0000_4900	0x0000_49FF
EPwm11Regs	EPWM_REGS	0x0000_4A00	0x0000_4AFF
EPwm12Regs	EPWM_REGS	0x0000_4B00	0x0000_4BFF
EPwmXbarRegs ⁽¹⁾	EPWM_XBAR_REGS	0x0000_7A00	0x0000_7A3F
TrigRegs ⁽¹⁾	TRIG_REGS	0x0000_7940	0x0000_794F

⁽¹⁾ Only available on CPU1.

13.5.2 EPWM_REGS Registers

Table 13-15 lists the memory-mapped registers for the EPWM_REGS. All register offset addresses not listed in Table 13-15 should be considered as reserved locations and the register contents should not be modified.

Table 13-15. EPWM_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	TBCTL	Time Base Control Register		Go
1h	TBCTL2	Time Base Control Register 2		Go
4h	TBCTR	Time Base Counter Register		Go
5h	TBSTS	Time Base Status Register		Go
8h	CMPCTL	Counter Compare Control Register		Go
9h	CMPCTL2	Counter Compare Control Register 2		Go
Ch	DBCTL	Dead-Band Generator Control Register		Go
Dh	DBCTL2	Dead-Band Generator Control Register 2		Go
10h	AQCTL	Action Qualifier Control Register		Go
11h	AQTSRCSEL	Action Qualifier Trigger Event Source Select Register		Go
14h	PCCTL	PWM Chopper Control Register		Go
20h	HRCNFG	HRPWM Configuration Register	EALLOW	Go
21h	HRPWR	HRPWM Power Register	EALLOW	Go
26h	HRMSTEP	HRPWM MEP Step Register	EALLOW	Go
27h	HRCNFG2	HRPWM Configuration 2 Register	EALLOW	Go
2Dh	HRPCTL	High Resolution Period Control Register	EALLOW	Go
34h	GLDCTL	Global PWM Load Control Register	EALLOW	Go
35h	GLDCFG	Global PWM Load Config Register	EALLOW	Go
38h	EPWMXLINK	EPWMx Link Register		Go
40h	AQCTLA	Action Qualifier Control Register For Output A		Go
41h	AQCTLA2	Additional Action Qualifier Control Register For Output A		Go
42h	AQCTLB	Action Qualifier Control Register For Output B		Go
43h	AQCTLB2	Additional Action Qualifier Control Register For Output B		Go
47h	AQSFRC	Action Qualifier Software Force Register		Go
49h	AQCSFRC	Action Qualifier Continuous S/W Force Register		Go
50h	DBREDHR	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		Go
51h	DBRED	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		Go
52h	DBFEDHR	Dead-Band Generator Falling Edge Delay High Resolution Register		Go
53h	DBFED	Dead-Band Generator Falling Edge Delay Count Register		Go
60h	TBPHS	Time Base Phase High		Go
62h	TBPRDHR	Time Base Period High Resolution Register		Go
63h	TBPRD	Time Base Period Register		Go
6Ah	CMPA	Counter Compare A Register		Go
6Ch	CMPB	Compare B Register		Go
6Fh	CMPC	Counter Compare C Register		Go
71h	CMPD	Counter Compare D Register		Go
74h	GLDCTL2	Global PWM Load Control Register 2	EALLOW	Go
80h	TZSEL	Trip Zone Select Register	EALLOW	Go

Table 13-15. EPWM_REGS Registers (continued)

Offset	Acronym	Register Name	Write Protection	Section
82h	TZDCSEL	Trip Zone Digital Comparator Select Register	EALLOW	Go
84h	TZCTL	Trip Zone Control Register	EALLOW	Go
85h	TZCTL2	Additional Trip Zone Control Register	EALLOW	Go
86h	TZCTLDCA	Trip Zone Control Register Digital Compare A	EALLOW	Go
87h	TZCTLDCB	Trip Zone Control Register Digital Compare B	EALLOW	Go
8Dh	TZEINT	Trip Zone Enable Interrupt Register	EALLOW	Go
93h	TZFLG	Trip Zone Flag Register		Go
94h	TZCBCFLG	Trip Zone CBC Flag Register		Go
95h	TZOSTFLG	Trip Zone OST Flag Register		Go
97h	TZCLR	Trip Zone Clear Register	EALLOW	Go
98h	TZCBCCLR	Trip Zone CBC Clear Register	EALLOW	Go
99h	TZOSTCLR	Trip Zone OST Clear Register	EALLOW	Go
9Bh	TZFRC	Trip Zone Force Register	EALLOW	Go
A4h	ETSEL	Event Trigger Selection Register		Go
A6h	ETPS	Event Trigger Pre-Scale Register		Go
A8h	ETFLG	Event Trigger Flag Register		Go
AAh	ETCLR	Event Trigger Clear Register		Go
ACH	ETFRC	Event Trigger Force Register		Go
Aeh	ETINTPS	Event-Trigger Interrupt Pre-Scale Register		Go
B0h	ETSOCPS	Event-Trigger SOC Pre-Scale Register		Go
B2h	ETCNTINITCTL	Event-Trigger Counter Initialization Control Register		Go
B4h	ETCNTINIT	Event-Trigger Counter Initialization Register		Go
C0h	DCTRIPSEL	Digital Compare Trip Select Register	EALLOW	Go
C3h	DCACTL	Digital Compare A Control Register	EALLOW	Go
C4h	DCBCTL	Digital Compare B Control Register	EALLOW	Go
C7h	DCFCTL	Digital Compare Filter Control Register	EALLOW	Go
C8h	DCCAPCTL	Digital Compare Capture Control Register	EALLOW	Go
C9h	DCFOFFSET	Digital Compare Filter Offset Register		Go
CAh	DCFOFFSETCNT	Digital Compare Filter Offset Counter Register		Go
CBh	DCFWINDOW	Digital Compare Filter Window Register		Go
CCh	DCFWINDOWCNT	Digital Compare Filter Window Counter Register		Go
CFh	DCCAP	Digital Compare Counter Capture Register		Go
D2h	DCAHTRIPSEL	Digital Compare AH Trip Select	EALLOW	Go
D3h	DCALTRIPSEL	Digital Compare AL Trip Select	EALLOW	Go
D4h	DCBHTRIPSEL	Digital Compare BH Trip Select	EALLOW	Go
D5h	DCBLTRIPSEL	Digital Compare BL Trip Select	EALLOW	Go

13.5.2.1 TBCTL Register (Offset = 0h) [reset = 83h]

TBCTL is shown in [Figure 13-78](#) and described in [Table 13-16](#).

Time Base Control Register

Figure 13-78. 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/W=1-0h	R/W-0h		R/W-0h	R/W-0h	R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-16. TBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	<p>Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events</p> <p>00: Stop after the next time-base counter increment or decrement</p> <p>01: Stop when counter completes a whole cycle:</p> <ul style="list-style-type: none"> - 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) <p>1x: Free run</p>
13	PHSDIR	R/W	0h	<p>Phase Direction Bit</p> <p>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..</p> <p>In the up-count and down-count modes this bit is ignored.</p> <p>0: Count down after the synchronization event.</p> <p>1: Count up after the synchronization event.</p>
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 (TBCLK = EPWMCLK/(HSPCLKDIV * CLKDIV):</p> <p>000: /1 (default on reset)</p> <p>001: /2</p> <p>010: /4</p> <p>011: /8</p> <p>100: /16</p> <p>101: /32</p> <p>110: /64</p> <p>111: /128</p>

Table 13-16. TBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</p> <p>$TBCLK = EPWMCLK / (HSPCLKDIV \times CLKDIV)$. This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.</p> <p>000: /1</p> <p>001: /2 (default on reset)</p> <p>010: /4</p> <p>011: /6</p> <p>100: /8</p> <p>101: /10</p> <p>110: /12</p> <p>111: /14</p>
6	SWFSYNC	R=0/W=1	0h	<p>Software Forced Sync Pulse</p> <p>0: Writing a 0 has no effect and reads always return a 0.</p> <p>1: Writing a 1 forces a one-time synchronization pulse to be generated.</p> <p>This event is ORed with the EPWMxSYNCl input of the ePWM module.</p> <p>SWFSYNC is valid (operates) only when EPWMxSYNCl is selected by SYNCOSSEL = 00.</p>
5-4	SYNCOSSEL	R/W	0h	<p>Sync Output Select</p> <p>00: EPWMxSYNC</p> <p>01: CTR = zero: Time-base counter equal to zero (TBCTR = 0x00)</p> <p>10: CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)</p> <p>11: Disable EPWMxSYNCO signal</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 TBCTL2[PRDLDSYNC] bit.</p> <p>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>
2	PHSEN	R/W	0h	<p>Counter Reg Load from Phase Reg Enable (CNTLDE)</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>
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>

13.5.2.2 TBCTL2 Register (Offset = 1h) [reset = 0h]

TBCTL2 is shown in [Figure 13-79](#) and described in [Table 13-17](#).

Time Base Control Register 2

Figure 13-79. 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	SELFCLRTRREM	RESERVED				
R/W-0h	R/W-0h	R/W-0h	R=0-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-17. 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[PRDLDD]=0.
13-12	SYNCOSELX	R/W	0h	Extended selection bits for SYNCOUT 00: Disabled EPWMxSYNCO sync signal 01: EPWMxSYNCO = CMPC 10: EPWMxSYNCO = CMPD 11: Reserved
11-8	RESERVED	R=0	0h	Reserved
7	OSHTSYNC	R/W	0h	Oneshot sync bit 0: Writing a '0' has no effect. 1: Allow one sync pulse to propagate.
6	OSHTSYNCMODE	R/W	0h	Oneshot sync enable bit 0: Oneshot sync mode disabled 1: Oneshot sync mode enabled
5	SELFCLRTRREM	R/W	0h	Loop back sync pulse to enable self sync operation 0: Self clear function of TRREM disabled. 1: Self clear function of TRREM enabled
4-0	RESERVED	R=0	0h	Reserved

13.5.2.3 TBCTR Register (Offset = 4h) [reset = 0h]

TBCTR is shown in [Figure 13-80](#) and described in [Table 13-18](#).

Time Base Counter Register

Figure 13-80. TBCTR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBCTR															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-18. TBCTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TBCTR	R/W	0h	Time Base Counter Register

13.5.2.4 TBSTS Register (Offset = 5h) [reset = 1h]

TBSTS is shown in [Figure 13-81](#) and described in [Table 13-19](#).

Time Base Status Register

Figure 13-81. TBSTS Register

15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED					CTRMX	SYNCl	CTDRlR
R=0-0h					R/W=1-0h	R/W=1-0h	R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-19. TBSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R=0	0h	Reserved
2	CTRMX	R/W=1	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.
1	SYNCl	R/W=1	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 (EPWMxSYNCl). Writing a 1 to this bit will clear the latched event.
0	CTDRlR	R	1h	Time Base Counter Direction Status Bit At reset, the counter is frozen therefore, this bit has no meaning. To make this bit meaningful, you must first set the appropriate mode via TBCTL[CTRMODE]. 0: Time-Base Counter is currently counting down. 1: Time-Base Counter is currently counting up.

13.5.2.5 CMPCTL Register (Offset = 8h) [reset = 0h]

CMPCTL is shown in [Figure 13-82](#) and described in [Table 13-20](#).

Counter Compare Control Register

Figure 13-82. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-20. 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.
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.
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
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
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

Table 13-20. CMPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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)
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)

13.5.2.6 CMPCTL2 Register (Offset = 9h) [reset = 0h]

CMPCTL2 is shown in [Figure 13-83](#) and described in [Table 13-21](#).

Counter Compare Control Register 2

Figure 13-83. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-21. 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 LOADCMODE 01: Shadow to Active Load of CMPD occurs both according to LOADCMODE 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.
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.
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.
5	RESERVED	R=0	0h	Reserved
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.

Table 13-21. CMPCTL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.

13.5.2.7 DBCTL Register (Offset = Ch) [reset = 0h]

DBCTL is shown in [Figure 13-84](#) and described in [Table 13-22](#).

Dead-Band Generator Control Register

Figure 13-84. 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	
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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-22. 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.
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 (IN-MODE 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.
13-12	OUTSWAP	R/W	0h	Dead Band Output Swap Control Bit 13 controls the S7 switch and bit 12 controls the S6 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).
11	SHDWDBFEDMODE	R/W	0h	FED Dead-Band Load Mode 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." 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).

Table 13-22. DBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	SHDWDBREDDMODE	R/W	0h	RED Dead-Band Load Mode 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." 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).
9-8	LOADFEDMODE	R/W	0h	Active DBFED 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.
7-6	LOADREDDMODE	R/W	0h	Active DBRED 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.
5-4	IN_MODE	R/W	0h	Dead-Band Input Mode Control 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. 00: EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay. 01: EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal. 10: EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal. 11: EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.
3-2	POLSEL	R/W	0h	Polarity Select Control 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. 00: Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default). 01: Active low complementary (ALC) mode. EPWMxA is inverted. 10: Active high complementary (AHC). EPWMxB is inverted. 11: Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.

Table 13-22. DBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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)</p> <p>Bpath = FED (Falling Edge Delay in B signal path)</p> <p>10: Apath = RED (Rising Edge Delay in A signal path)</p> <p>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>

13.5.2.8 DBCTL2 Register (Offset = Dh) [reset = 0h]

DBCTL2 is shown in [Figure 13-85](#) and described in [Table 13-23](#).

Dead-Band Generator Control Register 2

Figure 13-85. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-23. 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.
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

13.5.2.9 AQCTL Register (Offset = 10h) [reset = 0h]

AQCTL is shown in [Figure 13-86](#) and described in [Table 13-24](#).

Action Qualifier Control Register

Figure 13-86. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-24. 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.
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.
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.
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.

Table 13-24. 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.
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.

13.5.2.10 AQTSRCSEL Register (Offset = 11h) [reset = 0h]

AQTSRCSEL is shown in [Figure 13-87](#) and described in [Table 13-25](#).

Action Qualifier Trigger Event Source Select Register

Figure 13-87. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-25. 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: EPWMxSYNCIN 1xxx: Reserved
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: EPWMxSYNCIN 1xxx: Reserved

13.5.2.11 PCCTL Register (Offset = 14h) [reset = 0h]

PCCTL is shown in [Figure 13-88](#) and described in [Table 13-26](#).

PWM Chopper Control Register

Figure 13-88. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-26. 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
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)

Table 13-26. PCCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-1	OSHTWTH	R/W	0h	One-Shot Pulse Width 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)
0	CHPEN	R/W	0h	PWM-Chopping Enable 0: Disable (bypass) PWM chopping function 1: Enable chopping function

13.5.2.12 HRCNFG Register (Offset = 20h) [reset = 0h]

HRCNFG is shown in [Figure 13-89](#) and described in [Table 13-27](#).

HRPWM Configuration Register

Figure 13-89. HRCNFG Register

15	14	13	12	11	10	9	8
RESERVED		RESERVED	HRLOADB		CTLMODEB	EDGMODEB	
R-0h		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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-27. HRCNFG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	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
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).
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)
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.

Table 13-27. 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.</p> <p>1: Automatic HRMSTEP scaling is enabled.</p> <p>If application software is manually scaling the fractional duty cycle, or phase (i.e. software sets $CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor}) < 8 + 0x080$ for duty cycle), then this mode must be disabled.</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>0: ePWMxB output is normal.</p> <p>1: ePWMxB output is inverted version of ePWMxA signal.</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)</p> <p>01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)</p> <p>10: Load on either CTR = Zero or CTR = PRD</p> <p>11: Reserved</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)</p> <p>1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode).</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)</p> <p>01: MEP control of rising edge (CMPAHR)</p> <p>10: MEP control of falling edge (CMPAHR)</p> <p>11: MEP control of both edges (TBPHSHR or TBPRDHR)</p>

13.5.2.13 HRPWR Register (Offset = 21h) [reset = 0h]

HRPWR is shown in [Figure 13-90](#) and described in [Table 13-28](#).

HRPWM Power Register

Figure 13-90. HRPWR Register

15	14	13	12	11	10	9	8
CALPWRON	RESERVED					RESERVED	
R/W-0h	R=0-0h					R-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-28. 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
14-10	RESERVED	R=0	0h	Reserved
9-6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1-0	RESERVED	R	0h	Reserved

13.5.2.14 HRMSTEP Register (Offset = 26h) [reset = 0h]

HRMSTEP is shown in [Figure 13-91](#) and described in [Table 13-29](#).

HRPWM MEP Step Register

Figure 13-91. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-29. HRMSTEP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-0	HRMSTEP	R/W	0h	High Resolution MEP Step 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.

13.5.2.15 HRCNFG2 Register (Offset = 27h) [reset = 0h]

HRCNFG2 is shown in [Figure 13-92](#) and described in [Table 13-30](#).

HRPWM Configuration 2 Register

Figure 13-92. HRCNFG2 Register

15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED					
R-0h	R-0h	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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-30. HRCNFG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	RESERVED	R	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
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
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)

13.5.2.16 HRPCTL Register (Offset = 2Dh) [reset = 0h]

HRPCTL is shown in [Figure 13-93](#) and described in [Table 13-31](#).

High Resolution Period Control Register

Figure 13-93. HRPCTL Register

15	14	13	12	11	10	9	8
RESERVED							
R=0-0h							
7	6	5	4	3	2	1	0
RESERVED	PWMSYNCSSELX			RESERVED	TBPHSHRLOADE	PWMSYNCSSEL	HRPE
R=0-0h	R/W-0h			R-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-31. HRPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R=0	0h	Reserved
6-4	PWMSYNCSSELX	R/W	0h	PWMSYNCS Source Select Bit 000: PWMSYNCS is defined by TRCTL[PWMSYNCSSEL] - > default condition (compatible with previous EPWM versions) 001: Reserved 010: Reserved 011: Reserved 100: PWMSYNCS = CMPC_eq, Count direction Up 101: PWMSYNCS = CMPC_eq, Count direction Down 110: PWMSYNCS = CMPD_eq, Count direction Up 111: PWMSYNCS = CMPD_eq, Count direction Down
3	RESERVED	R	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[SWFSYNCS] 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[SWFSYNCS] or digital compare event: 1: Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNCS] 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[SWFSYNCS] 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.
1	PWMSYNCSSEL	R/W	0h	PWMSYNCS Source Select Bit: This bit selects the source for the PWMSYNCS signal. The PWMSYNCS signal is used by external modules (such as COMP+DAC) for syncing timing to the selected EPWM module:

Table 13-31. HRPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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>

13.5.2.17 GLDCTL Register (Offset = 34h) [reset = 0h]

GLDCTL is shown in [Figure 13-94](#) and described in [Table 13-32](#).

Global PWM Load Control Register

Figure 13-94. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-32. GLDCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R=0	0h	Reserved
12-10	GLDCNT	R	0h	Global Reload 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
9-7	GLDPRD	R/W	0h	Global Reload 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)
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)

Table 13-32. 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>
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>

13.5.2.18 GLDCFG Register (Offset = 35h) [reset = 0h]

GLDCFG is shown in [Figure 13-95](#) and described in [Table 13-33](#).

Global PWM Load Config Register

Figure 13-95. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-33. 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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1

Table 13-33. GLDCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1
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 reload configuration if this bit is set and GLDCTL(GLD)=1

13.5.2.19 EPWMXLINK Register (Offset = 38h) [reset = X]

EPWMXLINK is shown in [Figure 13-96](#) and described in [Table 13-34](#).

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 13-96. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-34. 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</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>
27-20	RESERVED	R=0	0h	Reserved

Table 13-34. 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</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>
15-12	CMPCLINK	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</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>

Table 13-34. EPWMXLINK Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-8	CMPBLINK	R/W	X	<p>CMPB_CMPBHR Link Bits</p> <p>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.</p> <p>0000: ePWM1</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>
7-4	CMPALINK	R/W	X	<p>CMPA_CMPAHR Link Bits</p> <p>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.</p> <p>0000: ePWM1</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>

Table 13-34. 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</p> <p>0001: ePWM2</p> <p>0010: ePWM3</p> <p>0011: ePWM4</p> <p>0100: ePWM5</p> <p>0101: ePWM6</p> <p>0110: ePWM7</p> <p>0111: ePWM8</p> <p>1000: ePWM9</p> <p>1001: ePWM10</p> <p>1010: ePWM11</p> <p>1011: ePWM12</p> <p>1100: Reserved</p> <p>...</p> <p>1111: Reserved</p>

13.5.2.20 AQCTLA Register (Offset = 40h) [reset = 0h]

AQCTLA is shown in [Figure 13-97](#) and described in [Table 13-35](#).

Action Qualifier Control Register For Output A

Figure 13-97. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-35. 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.
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.
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.
5-4	CAU	R/W	0h	Action When TBCTR = CMPA 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.

Table 13-35. AQCTLA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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)</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>
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)</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>

13.5.2.21 AQCTLA2 Register (Offset = 41h) [reset = 0h]

AQCTLA2 is shown in [Figure 13-98](#) and described in [Table 13-36](#).

Additional Action Qualifier Control Register For Output A

Figure 13-98. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-36. AQCTLA2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-6	T2D	R/W	0h	Action when event occurs on T2 in 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.
5-4	T2U	R/W	0h	Action when event occurs on T2 in 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.
3-2	T1D	R/W	0h	Action when event occurs on T1 in 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.
1-0	T1U	R/W	0h	Action when event occurs on T1 in 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.

13.5.2.22 AQCTLB Register (Offset = 42h) [reset = 0h]

AQCTLB is shown in [Figure 13-99](#) and described in [Table 13-37](#).

Action Qualifier Control Register For Output B

Figure 13-99. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-37. 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.
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.
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.
5-4	CAU	R/W	0h	Action When TBCTR = CMPA 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.

Table 13-37. AQCTLB Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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)</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>
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)</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>

13.5.2.23 AQCTLB2 Register (Offset = 43h) [reset = 0h]

AQCTLB2 is shown in [Figure 13-100](#) and described in [Table 13-38](#).

Additional Action Qualifier Control Register For Output B

Figure 13-100. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-38. AQCTLB2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-6	T2D	R/W	0h	Action when event occurs on T2 in 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.
5-4	T2U	R/W	0h	Action when event occurs on T2 in 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.
3-2	T1D	R/W	0h	Action when event occurs on T1 in 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.
1-0	T1U	R/W	0h	Action when event occurs on T1 in 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.

13.5.2.24 AQSFRFC Register (Offset = 47h) [reset = 0h]

AQSFRFC is shown in [Figure 13-101](#) and described in [Table 13-39](#).

Action Qualifier Software Force Register

Figure 13-101. AQSFRFC 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	ACTSFA	
R/W-0h		R=0/W=1-0h		R/W-0h		R=0/W=1-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-39. AQSFRFC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7-6	RLDCSF	R/W	0h	AQCSFRFC 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).
5	OTSFB	R=0/W=1	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
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)
2	OTSFA	R=0/W=1	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 B. 1: Initiates a single software forced event
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)

13.5.2.25 AQCSFRC Register (Offset = 49h) [reset = 0h]

AQCSFRC is shown in [Figure 13-102](#) and described in [Table 13-40](#).

Action Qualifier Continuous S/W Force Register

Figure 13-102. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-40. AQCSFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R=0	0h	Reserved
3-2	CSFB	R/W	0h	Continuous Software Force on Output B 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]. 00: Forcing disabled, i.e., 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
1-0	CSFA	R/W	0h	Continuous Software Force on Output A 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. 00: Forcing disabled, i.e., 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

13.5.2.26 DBREDHR Register (Offset = 50h) [reset = 0h]

DBREDHR is shown in [Figure 13-103](#) and described in [Table 13-41](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

Figure 13-103. 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							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-41. DBREDHR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	DBREDHR	R/W	0h	Dead Band Rising Edge Delay High Resolution Bits
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

13.5.2.27 DBRED Register (Offset = 51h) [reset = 0h]

DBRED is shown in [Figure 13-104](#) and described in [Table 13-42](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

Figure 13-104. DBRED Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBRED															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-42. DBRED Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DBRED	R/W	0h	Rising edge delay value

13.5.2.28 DBFEDHR Register (Offset = 52h) [reset = 0h]

DBFEDHR is shown in [Figure 13-105](#) and described in [Table 13-43](#).

Dead-Band Generator Falling Edge Delay High Resolution Register

Figure 13-105. 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							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-43. DBFEDHR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-9	DBFEDHR	R/W	0h	Dead Band Falling Edge Delay High Resolution Bits
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

13.5.2.29 DBFED Register (Offset = 53h) [reset = 0h]

DBFED is shown in [Figure 13-106](#) and described in [Table 13-44](#).

Dead-Band Generator Falling Edge Delay Count Register

Figure 13-106. DBFED Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBFED															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-44. DBFED Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DBFED	R/W	0h	Falling Edge Delay Count 14-bit counter

13.5.2.30 TBPHS Register (Offset = 60h) [reset = 0h]

TBPHS is shown in [Figure 13-107](#) and described in [Table 13-45](#).

Time Base Phase High

Figure 13-107. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-45. TBPHS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	TBPHS	R/W	0h	<p>Phase Offset Register</p> <p>These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal.</p> <ul style="list-style-type: none"> - 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 (EPWMxSYNCI) or by a software forced synchronization.
15-0	TBPHSHR	R/W	0h	Phase Offset (High Resolution) Register

13.5.2.31 TBPRDHR Register (Offset = 62h) [reset = 0h]

TBPRDHR is shown in [Figure 13-108](#) and described in [Table 13-46](#).

Time Base Period High Resolution Register

Figure 13-108. TBPRDHR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBPRDHR															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-46. TBPRDHR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TBPRDHR	R/W	0h	<p>Period High Resolution Bits</p> <p>These 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDLD] 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>

13.5.2.32 TBPRD Register (Offset = 63h) [reset = 0h]

TBPRD is shown in [Figure 13-109](#) and described in [Table 13-47](#).

Time Base Period Register

Figure 13-109. TBPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBPRD															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-47. 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[PRDL] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> - If TBCTL[PRDL] = 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. - If TBCTL[PRDL] = 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. - The active and shadow registers share the same memory map address.

13.5.2.33 CMPA Register (Offset = 6Ah) [reset = 0h]

CMPA is shown in [Figure 13-110](#) and described in [Table 13-48](#).

Counter Compare A Register

Figure 13-110. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-48. 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"> - Do nothing <p>the event is ignored.</p> <ul style="list-style-type: none"> - Clear: Pull the EPWMxA and/or EPWMxB signal low - Set: Pull the EPWMxA and/or EPWMxB signal high - Toggle the EPWMxA and/or EPWMxB signal <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"> - 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. - Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full. - 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. - In either mode, the active and shadow registers share the same memory map address.
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>

13.5.2.34 CMPB Register (Offset = 6Ch) [reset = 0h]

CMPB is shown in [Figure 13-111](#) and described in [Table 13-49](#).

Compare B Register

Figure 13-111. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-49. 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"> - Do nothing <p>the event is ignored.</p> <ul style="list-style-type: none"> - Clear: Pull the EPWMxA and/or EPWMxB signal low - Set: Pull the EPWMxA and/or EPWMxB signal high - Toggle the EPWMxA and/or EPWMxB signal <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"> - 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. - Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full. - 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. - In either mode, the active and shadow registers share the same memory map address.
15-0	CMPBHR	R/W	0h	Compare B High Resolution Bits

13.5.2.35 CMPC Register (Offset = 6Fh) [reset = 0h]

CMPC is shown in [Figure 13-112](#) and described in [Table 13-50](#).

Counter Compare C Register

Figure 13-112. CMPC Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPC															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-50. 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"> - 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: - 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. - In either mode, the active and shadow registers share the same memory map address.

13.5.2.36 CMPD Register (Offset = 71h) [reset = 0h]

CMPD is shown in [Figure 13-113](#) and described in [Table 13-51](#).

Counter Compare D Register

Figure 13-113. CMPD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPD															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-51. 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"> - 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: - 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. - In either mode, the active and shadow registers share the same memory map address.

13.5.2.37 GLDCTL2 Register (Offset = 74h) [reset = 0h]

GLDCTL2 is shown in [Figure 13-114](#) and described in [Table 13-52](#).

Global PWM Load Control Register 2

Figure 13-114. 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/W=1-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-52. GLDCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R=0	0h	Reserved
1	GFRCLD	R=0/W=1	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.
0	OSHTLD	R=0/W=1	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.

13.5.2.38 TZSEL Register (Offset = 80h) [reset = 0h]

TZSEL is shown in [Figure 13-115](#) and described in [Table 13-53](#).

Trip Zone Select Register

Figure 13-115. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-53. 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.
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.
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
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
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
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
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
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
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
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
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

Table 13-53. TZSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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
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
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
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

13.5.2.39 TZDCSEL Register (Offset = 82h) [reset = 0h]

TZDCSEL is shown in [Figure 13-116](#) and described in [Table 13-54](#).

Trip Zone Digital Comparator Select Register

Figure 13-116. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-54. 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
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
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

Table 13-54. 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

13.5.2.40 TZCTL Register (Offset = 84h) [reset = 0h]

TZCTL is shown in [Figure 13-117](#) and described in [Table 13-55](#).

Trip Zone Control Register

Figure 13-117. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-55. 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
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
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
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
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.

Table 13-55. 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)</p> <p>01: Force EPWMxA to a high state</p> <p>10: Force EPWMxA to a low state</p> <p>11: Do nothing, no action is taken on EPWMxA.</p>

13.5.2.41 TZCTL2 Register (Offset = 85h) [reset = 0h]

TZCTL2 is shown in [Figure 13-118](#) and described in [Table 13-56](#).

Additional Trip Zone Control Register

Figure 13-118. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-56. 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, TZCTLDCA and TZCTLDCB. Settings in TZCTL are ignored
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
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
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

Table 13-56. TZCTL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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

13.5.2.42 TZCTLDCA Register (Offset = 86h) [reset = 0h]

TZCTLDCA is shown in [Figure 13-119](#) and described in [Table 13-57](#).

Trip Zone Control Register Digital Compare A

Figure 13-119. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-57. 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
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
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

Table 13-57. TZCTLDCR 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

13.5.2.43 TZCTLDCB Register (Offset = 87h) [reset = 0h]

TZCTLDCB is shown in [Figure 13-120](#) and described in [Table 13-58](#).

Trip Zone Control Register Digital Compare B

Figure 13-120. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-58. 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
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
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

Table 13-58. 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

13.5.2.44 TZEINT Register (Offset = 8Dh) [reset = 0h]

TZEINT is shown in [Figure 13-121](#) and described in [Table 13-59](#).

Trip Zone Enable Interrupt Register

Figure 13-121. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-59. 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
5	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Interrupt Enable 0: Disabled 1: Enabled
4	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Interrupt Enable 0: Disabled 1: Enabled
3	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Interrupt Enable 0: Disabled 1: Enabled
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.
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.
0	RESERVED	R=0	0h	Reserved

13.5.2.45 TZFLG Register (Offset = 93h) [reset = 0h]

TZFLG is shown in [Figure 13-122](#) and described in [Table 13-60](#).

Trip Zone Flag Register

Figure 13-122. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-60. 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
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
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
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
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.
1	CBC	R	0h	Latched Status Flag for Cycle-By-Cycle Trip Event 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 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. This bit is cleared by writing the appropriate value to the TZCLR register.

Table 13-60. TZFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INT	R	0h	<p>Latched Trip Interrupt Status Flag</p> <p>0: Indicates no interrupt has been generated.</p> <p>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.</p>

13.5.2.46 TZCBCFLG Register (Offset = 94h) [reset = 0h]

TZCBCFLG is shown in [Figure 13-123](#) and described in [Table 13-61](#).

Trip Zone CBC Flag Register

Figure 13-123. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-61. 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.
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.
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.
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.
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.
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.
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.
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.

13.5.2.47 TZOSTFLG Register (Offset = 95h) [reset = 0h]

TZOSTFLG is shown in [Figure 13-124](#) and described in [Table 13-62](#).

Trip Zone OST Flag Register

Figure 13-124. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-62. 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.
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.
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.
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.
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.
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.
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.
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.

13.5.2.48 TZCLR Register (Offset = 97h) [reset = 0h]

TZCLR is shown in [Figure 13-125](#) and described in [Table 13-63](#).

Trip Zone Clear Register

Figure 13-125. 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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-63. 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: Reserved (CBC trip latch is not cleared)
13-7	RESERVED	R=0	0h	Reserved
6	DCBEVT2	R=0/W=1	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.
5	DCBEVT1	R=0/W=1	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.
4	DCAEVT2	R=0/W=1	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.
3	DCAEVT1	R=0/W=1	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.
2	OST	R=0/W=1	0h	Clear Flag for One-Shot Trip (OST) Latch 0: Has no effect. Always reads back a 0. 1: Clears this Trip (set) condition.
1	CBC	R=0/W=1	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.
0	INT	R=0/W=1	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.

13.5.2.49 TZCBCCLR Register (Offset = 98h) [reset = 0h]

TZCBCCLR is shown in [Figure 13-126](#) and described in [Table 13-64](#).

Trip Zone CBC Clear Register

Figure 13-126. 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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-64. TZCBCCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7	DCBEVT2	R=0/W=1	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.
6	DCAEVT2	R=0/W=1	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.
5	CBC6	R=0/W=1	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.
4	CBC5	R=0/W=1	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.
3	CBC4	R=0/W=1	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.
2	CBC3	R=0/W=1	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.
1	CBC2	R=0/W=1	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.
0	CBC1	R=0/W=1	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.

13.5.2.50 TZOSTCLR Register (Offset = 99h) [reset = 0h]

TZOSTCLR is shown in [Figure 13-127](#) and described in [Table 13-65](#).

Trip Zone OST Clear Register

Figure 13-127. 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/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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-65. TZOSTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R=0	0h	Reserved
7	DCBEVT1	R=0/W=1	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.
6	DCAEVT1	R=0/W=1	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.
5	OST6	R=0/W=1	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.
4	OST5	R=0/W=1	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.
3	OST4	R=0/W=1	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.
2	OST3	R=0/W=1	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.
1	OST2	R=0/W=1	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.
0	OST1	R=0/W=1	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.

13.5.2.51 TZFRC Register (Offset = 9Bh) [reset = 0h]

TZFRC is shown in [Figure 13-128](#) and described in [Table 13-66](#).

Trip Zone Force Register

Figure 13-128. 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/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=0-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-66. TZFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R=0	0h	Reserved
6	DCBEVT2	R=0/W=1	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.
5	DCBEVT1	R=0/W=1	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.
4	DCAEVT2	R=0/W=1	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.
3	DCAEVT1	R=0/W=1	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.
2	OST	R=0/W=1	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.
1	CBC	R=0/W=1	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.
0	RESERVED	R=0	0h	Reserved

13.5.2.52 ETSEL Register (Offset = A4h) [reset = 0h]

ETSEL is shown in [Figure 13-129](#) and described in [Table 13-67](#).

Event Trigger Selection Register

Figure 13-129. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-67. 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.
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.
11	SOCAEN	R/W	0h	Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse 0: Disable EPWMxSOCA. 1: Enable EPWMxSOCA pulse.

Table 13-67. ETSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10-8	SOCASEL	R/W	0h	<p>EPWMxSOCA Selection Options</p> <p>These bits determine when a EPWMxSOCA pulse will be generated.</p> <p>000: Enable DCAEVT1.soc event</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 SOCASELCMP bit.</p>
7	RESERVED	R=0	0h	Reserved
6	INTSELCMP	R/W	0h	<p>EPWMxINT 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 INTSEL 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 INTSEL selection mux.</p>
5	SOCBSELCMP	R/W	0h	<p>EPWMxSOCB 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 SOCBSEL 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 SOCBSEL selection mux.</p>
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>

Table 13-67. ETSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INTEN	R/W	0h	Enable ePWM Interrupt (EPWMx_INT) Generation 0: Disable EPWMx_INT generation 1: Enable EPWMx_INT generation
2-0	INTSEL	R/W	0h	ePWM Interrupt (EPWMx_INT) Selection Options 000: Reserved 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 INTSELCMP bit.

13.5.2.53 ETPS Register (Offset = A6h) [reset = 0h]

ETPS is shown in [Figure 13-130](#) and described in [Table 13-68](#).

Event Trigger Pre-Scale Register

Figure 13-130. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-68. 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.
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
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.

Table 13-68. 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[SOCASEN] = 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>
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>
4	INTPSEL	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>
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>

Table 13-68. 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>

13.5.2.54 ETFLG Register (Offset = A8h) [reset = 0h]

ETFLG is shown in [Figure 13-131](#) and described in [Table 13-69](#).

Event Trigger Flag Register

Figure 13-131. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-69. 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.
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.
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.

13.5.2.55 ETCLR Register (Offset = AAh) [reset = 0h]

ETCLR is shown in [Figure 13-132](#) and described in [Table 13-70](#).

Event Trigger Clear Register

Figure 13-132. 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/W=1-0h	R=0/W=1-0h	R=0-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-70. ETCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R=0	0h	Reserved
3	SOCB	R=0/W=1	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
2	SOCA	R=0/W=1	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
1	RESERVED	R=0	0h	Reserved
0	INT	R=0/W=1	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

13.5.2.56 ETFRC Register (Offset = ACh) [reset = 0h]

ETFRC is shown in [Figure 13-133](#) and described in [Table 13-71](#).

Event Trigger Force Register

Figure 13-133. 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/W=1-0h	R=0/W=1-0h	R=0-0h	R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-71. ETFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R=0	0h	Reserved
3	SOCB	R=0/W=1	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>
2	SOCA	R=0/W=1	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>
1	RESERVED	R=0	0h	Reserved
0	INT	R=0/W=1	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>

13.5.2.57 ETINTPS Register (Offset = AEh) [reset = 0h]

ETINTPS is shown in [Figure 13-134](#) and described in [Table 13-72](#).

Event-Trigger Interrupt Pre-Scale Register

Figure 13-134. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-72. 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[INTPSSEL]=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
3-0	INTPRD2	R/W	0h	EPWMxINT Period 2 Select When ETPS[INTPSSEL] = 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)

13.5.2.58 ETSOCPS Register (Offset = B0h) [reset = 0h]

ETSOCPS is shown in [Figure 13-135](#) and described in [Table 13-73](#).

Event-Trigger SOC Pre-Scale Register

Figure 13-135. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-73. ETSOCPS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	SOCBCNT2	R	0h	EPWMxSOCB Counter 2 When ETPS[SOCPSSEL] = 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
11-8	SOCBPRD2	R/W	0h	EPWMxSOCB Period 2 Select When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCB pulse is generated: 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)
7-4	SOCACNT2	R	0h	EPWMxSOCA Counter 2 When ETPS[SOCPSSEL] = 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

Table 13-73. 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</p> <p>0001: Generate interrupt on SOCACNT2 = 1 (first event)</p> <p>0010: Generate interrupt on SOCACNT2 = 2 (second event)</p> <p>0011: Generate interrupt on SOCACNT2 = 3 (third event)</p> <p>0100: Generate interrupt on SOCACNT2 = 4 (fourth event)</p> <p>...</p> <p>1111: Generate interrupt on SOCACNT2 = 15 (fifteenth event)</p>

13.5.2.59 ETCNTINITCTL Register (Offset = B2h) [reset = 0h]

ETCNTINITCTL is shown in [Figure 13-136](#) and described in [Table 13-74](#).

Event-Trigger Counter Initialization Control Register

Figure 13-136. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-74. 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.
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.
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.
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].
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].
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].
9-0	RESERVED	R=0	0h	Reserved

13.5.2.60 ETCNTINIT Register (Offset = B4h) [reset = 0h]

ETCNTINIT is shown in [Figure 13-137](#) and described in [Table 13-75](#).

Event-Trigger Counter Initialization Register

Figure 13-137. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-75. 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.
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.
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.

13.5.2.61 DCTRIPSEL Register (Offset = C0h) [reset = 0h]

DCTRIPSEL is shown in [Figure 13-138](#) and described in [Table 13-76](#).

Digital Compare Trip Select Register

Figure 13-138. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-76. 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)
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)
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)

Table 13-76. DCTRIPSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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)

13.5.2.62 DCACTL Register (Offset = C3h) [reset = 0h]

DCACTL is shown in [Figure 13-139](#) and described in [Table 13-77](#).

Digital Compare A Control Register

Figure 13-139. 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
RESERVED				EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-77. DCACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R=0	0h	Reserved
9	EVT2FRCSYNCSSEL	R/W	0h	DCAEVT2 Force Synchronization Signal Select 0: Source Is Synchronous Signal 1: Source Is Asynchronous Signal
8	EVT2SRCSEL	R/W	0h	DCAEVT2 Source Signal Select 0: Source Is DCAEVT2 Signal 1: Source Is DCEVTFILT Signal
7-4	RESERVED	R=0	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCAEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled
2	EVT1SOCE	R/W	0h	DCAEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled
1	EVT1FRCSYNCSSEL	R/W	0h	DCAEVT1 Force Synchronization Signal Select 0: Source Is Synchronous Signal 1: Source Is Asynchronous Signal
0	EVT1SRCSEL	R/W	0h	DCAEVT1 Source Signal Select 0: Source Is DCAEVT1 Signal 1: Source Is DCEVTFILT Signal

13.5.2.63 DCBCTL Register (Offset = C4h) [reset = 0h]

DCBCTL is shown in [Figure 13-140](#) and described in [Table 13-78](#).

Digital Compare B Control Register

Figure 13-140. 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
RESERVED				EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-78. DCBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R=0	0h	Reserved
9	EVT2FRCSYNCSSEL	R/W	0h	DCBEVT2 Force Synchronization Signal Select 0: Source Is Synchronous Signal 1: Source Is Asynchronous Signal
8	EVT2SRCSEL	R/W	0h	DCBEVT2 Source Signal Select 0: Source Is DCBEVT2 Signal 1: Source Is DCEVTFILT Signal
7-4	RESERVED	R=0	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCBEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled
2	EVT1SOCE	R/W	0h	DCBEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled
1	EVT1FRCSYNCSSEL	R/W	0h	DCBEVT1 Force Synchronization Signal Select 0: Source Is Synchronous Signal 1: Source Is Asynchronous Signal
0	EVT1SRCSEL	R/W	0h	DCBEVT1 Source Signal Select 0: Source Is DCBEVT1 Signal 1: Source Is DCEVTFILT Signal

13.5.2.64 DCFCTL Register (Offset = C7h) [reset = 0h]

DCFCTL is shown in [Figure 13-141](#) and described in [Table 13-79](#).

Digital Compare Filter Control Register

Figure 13-141. DCFCTL Register

15	14	13	12	11	10	9	8
RESERVED			RESERVED			RESERVED	
R-0h			R-0h			R-0h	
7	6	5	4	3	2	1	0
RESERVED	RESERVED	PULSESEL		BLANKINV	BLANKE	SRCSEL	
R=0-0h	R-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-79. DCFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-10	RESERVED	R	0h	Reserved
9-8	RESERVED	R	0h	Reserved
7	RESERVED	R=0	0h	Reserved
6	RESERVED	R	0h	Reserved
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
3	BLANKINV	R/W	0h	Blanking Window Inversion 0: Blanking window not inverted 1: Blanking window inverted
2	BLANKE	R/W	0h	Blanking Window Enable/Disable 0: Blanking window is disabled 1: Blanking window is enabled
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

13.5.2.65 DCCAPCTL Register (Offset = C8h) [reset = 0h]

DCCAPCTL is shown in [Figure 13-142](#) and described in [Table 13-80](#).

Digital Compare Capture Control Register

Figure 13-142. DCCAPCTL Register

15	14	13	12	11	10	9	8
CAPMODE	CAPCLR	CAPSTS	RESERVED				
R/W-0h	R=0/W=1-0h	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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-80. DCCAPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	CAPMODE	R/W	0h	Counter Capture Mode 0: DCCAP function is controlled by PULSESEL bit in the DCFCTL register (compatible with previous EPWM versions) 1: Type-4 addition, DCCAP function is controlled by software
14	CAPCLR	R=0/W=1	0h	DC Capture Latched Status Clear Flag 0: Writing a 0 has no effect. 1: Writing a 1 will clear this CAPSTS (set) condition.
13	CAPSTS	R	0h	Latched Status Flag for Capture Event 0: No DC capture event occurred. 1: A DC capture event has occurred.
12-2	RESERVED	R=0	0h	Reserved
1	SHDWMODE	R/W	0h	TBCTR Counter Capture Shadow Select Mode 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. 1: Active Mode. In this mode the shadow register is disabled. CPU reads from the DCCAP register will always return the active register contents.
0	CAPE	R/W	0h	TBCTR Counter Capture Enable/Disable 0: Disable the time-base counter capture. 1: Enable the time-base counter capture.

13.5.2.66 DCFOFFSET Register (Offset = C9h) [reset = 0h]

DCFOFFSET is shown in [Figure 13-143](#) and described in [Table 13-81](#).

Digital Compare Filter Offset Register

Figure 13-143. DCFOFFSET Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCFOFFSET															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-81. 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>

13.5.2.67 DCFFSETCNT Register (Offset = CAh) [reset = 0h]

DCFFSETCNT is shown in [Figure 13-144](#) and described in [Table 13-82](#).

Digital Compare Filter Offset Counter Register

Figure 13-144. DCFFSETCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCFFSETCNT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-82. DCFFSETCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DCFFSETCNT	R	0h	Blanking Offset Counter 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.

13.5.2.68 DCFWINDOW Register (Offset = CBh) [reset = 0h]

DCFWINDOW is shown in [Figure 13-145](#) and described in [Table 13-83](#).

Digital Compare Filter Window Register

Figure 13-145. DCFWINDOW Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCFWINDOW															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-83. 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-FFh: 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 restarted. The blanking window can cross a PWM period boundary.</p>

13.5.2.69 DCFWINDOWCNT Register (Offset = CCh) [reset = 0h]

DCFWINDOWCNT is shown in [Figure 13-146](#) and described in [Table 13-84](#).

Digital Compare Filter Window Counter Register

Figure 13-146. DCFWINDOWCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCFWINDOWCNT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-84. DCFWINDOWCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DCFWINDOWCNT	R	0h	Blanking Window Counter These 8 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.

13.5.2.70 DCCAP Register (Offset = CFh) [reset = 0h]

DCCAP is shown in [Figure 13-147](#) and described in [Table 13-85](#).

Digital Compare Counter Capture Register

Figure 13-147. DCCAP Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCCAP															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-85. 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"> - 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. - 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.

13.5.2.71 DCAHTRIPSEL Register (Offset = D2h) [reset = 0h]

DCAHTRIPSEL is shown in [Figure 13-148](#) and described in [Table 13-86](#).

Digital Compare AH Trip Select

Figure 13-148. 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-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-86. 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
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
12	RESERVED	R	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
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
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
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
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
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
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
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

Table 13-86. DCAHTRIPSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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
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
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

13.5.2.72 DCALTRIPSEL Register (Offset = D3h) [reset = 0h]

DCALTRIPSEL is shown in [Figure 13-149](#) and described in [Table 13-87](#).

Digital Compare AL Trip Select

Figure 13-149. 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-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-87. 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
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
12	RESERVED	R	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
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
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
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
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
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
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
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

Table 13-87. DCALTRIPSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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
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
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

13.5.2.73 DCBHTRIPSEL Register (Offset = D4h) [reset = 0h]

DCBHTRIPSEL is shown in [Figure 13-150](#) and described in [Table 13-88](#).

Digital Compare BH Trip Select

Figure 13-150. 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-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-88. 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
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
12	RESERVED	R	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
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
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
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
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
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
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
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

Table 13-88. DCBHTRIPSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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
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
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

13.5.2.74 DCBLTRIPSEL Register (Offset = D5h) [reset = 0h]

DCBLTRIPSEL is shown in [Figure 13-151](#) and described in [Table 13-89](#).

Digital Compare BL Trip Select

Figure 13-151. 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-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-89. 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
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
12	RESERVED	R	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
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
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
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
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
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
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
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

Table 13-89. DCBLTRIPSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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
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
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

13.5.3 EPWM_XBAR_REGS Registers

Table 13-90 lists the memory-mapped registers for the EPWM_XBAR_REGS. All register offset addresses not listed in Table 13-90 should be considered as reserved locations and the register contents should not be modified.

Table 13-90. EPWM_XBAR_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	TRIP4MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	Go
2h	TRIP4MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	Go
4h	TRIP5MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	Go
6h	TRIP5MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	Go
8h	TRIP7MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	Go
Ah	TRIP7MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	Go
Ch	TRIP8MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	Go
Eh	TRIP8MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	Go
10h	TRIP9MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	Go
12h	TRIP9MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	Go
14h	TRIP10MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	Go
16h	TRIP10MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	Go
18h	TRIP11MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	Go
1Ah	TRIP11MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	Go
1Ch	TRIP12MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	Go
1Eh	TRIP12MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	Go
20h	TRIP4MUXENABLE	ePWM XBAR Mux Enable for TRIP4	EALLOW	Go
22h	TRIP5MUXENABLE	ePWM XBAR Mux Enable for TRIP5	EALLOW	Go
24h	TRIP7MUXENABLE	ePWM XBAR Mux Enable for TRIP7	EALLOW	Go
26h	TRIP8MUXENABLE	ePWM XBAR Mux Enable for TRIP8	EALLOW	Go
28h	TRIP9MUXENABLE	ePWM XBAR Mux Enable for TRIP9	EALLOW	Go
2Ah	TRIP10MUXENABLE	ePWM XBAR Mux Enable for TRIP10	EALLOW	Go
2Ch	TRIP11MUXENABLE	ePWM XBAR Mux Enable for TRIP11	EALLOW	Go
2Eh	TRIP12MUXENABLE	ePWM XBAR Mux Enable for TRIP12	EALLOW	Go
38h	TRIPOUTINV	ePWM XBAR Output Inversion Register	EALLOW	Go
3Eh	TRIPLOCK	ePWM XBAR Configuration Lock register	EALLOW	Go

13.5.3.1 TRIP4MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

TRIP4MUX0TO15CFG is shown in [Figure 13-152](#) and described in [Table 13-91](#).

ePWM XBAR Mux Configuration for TRIP4

Figure 13-152. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-91. TRIP4MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.

Table 13-91. TRIP4MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.

Table 13-91. TRIP4MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP4 of EPWM-XBAR 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.

13.5.3.2 TRIP4MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

TRIP4MUX16TO31CFG is shown in [Figure 13-153](#) and described in [Table 13-92](#).

ePWM XBAR Mux Configuration for TRIP4

Figure 13-153. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-92. TRIP4MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.

Table 13-92. TRIP4MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.

Table 13-92. TRIP4MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP4 of EPWM-XBAR 00 : Select .1 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.

13.5.3.3 TRIP5MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

TRIP5MUX0TO15CFG is shown in [Figure 13-154](#) and described in [Table 13-93](#).

ePWM XBAR Mux Configuration for TRIP5

Figure 13-154. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-93. TRIP5MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

Table 13-93. TRIP5MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

Table 13-93. TRIP5MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

13.5.3.4 TRIP5MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

TRIP5MUX16TO31CFG is shown in [Figure 13-155](#) and described in [Table 13-94](#).

ePWM XBAR Mux Configuration for TRIP5

Figure 13-155. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-94. TRIP5MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

Table 13-94. TRIP5MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

Table 13-94. TRIP5MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP5 of EPWM-XBAR 00 : Select .1 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.

13.5.3.5 TRIP7MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

TRIP7MUX0TO15CFG is shown in [Figure 13-156](#) and described in [Table 13-95](#).

ePWM XBAR Mux Configuration for TRIP7

Figure 13-156. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-95. TRIP7MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

Table 13-95. TRIP7MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

Table 13-95. TRIP7MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

13.5.3.6 TRIP7MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

TRIP7MUX16TO31CFG is shown in [Figure 13-157](#) and described in [Table 13-96](#).

ePWM XBAR Mux Configuration for TRIP7

Figure 13-157. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-96. TRIP7MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

Table 13-96. TRIP7MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

Table 13-96. TRIP7MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP7 of EPWM-XBAR 00 : Select .1 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.

13.5.3.7 TRIP8MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

TRIP8MUX0TO15CFG is shown in [Figure 13-158](#) and described in [Table 13-97](#).

ePWM XBAR Mux Configuration for TRIP8

Figure 13-158. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-97. TRIP8MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

Table 13-97. TRIP8MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

Table 13-97. TRIP8MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

13.5.3.8 TRIP8MUX16TO31CFG Register (Offset = Eh) [reset = 0h]

TRIP8MUX16TO31CFG is shown in [Figure 13-159](#) and described in [Table 13-98](#).

ePWM XBAR Mux Configuration for TRIP8

Figure 13-159. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-98. TRIP8MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

Table 13-98. TRIP8MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

Table 13-98. TRIP8MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP8 of EPWM-XBAR 00 : Select .1 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.

13.5.3.9 TRIP9MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

TRIP9MUX0TO15CFG is shown in [Figure 13-160](#) and described in [Table 13-99](#).

ePWM XBAR Mux Configuration for TRIP9

Figure 13-160. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-99. TRIP9MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

Table 13-99. TRIP9MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

Table 13-99. TRIP9MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

13.5.3.10 TRIP9MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

TRIP9MUX16TO31CFG is shown in [Figure 13-161](#) and described in [Table 13-100](#).

ePWM XBAR Mux Configuration for TRIP9

Figure 13-161. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-100. TRIP9MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

Table 13-100. TRIP9MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

Table 13-100. TRIP9MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP9 of EPWM-XBAR 00 : Select .1 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.

13.5.3.11 TRIP10MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

TRIP10MUX0TO15CFG is shown in [Figure 13-162](#) and described in [Table 13-101](#).

ePWM XBAR Mux Configuration for TRIP10

Figure 13-162. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-101. TRIP10MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

Table 13-101. TRIP10MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

Table 13-101. TRIP10MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

13.5.3.12 TRIP10MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

TRIP10MUX16TO31CFG is shown in [Figure 13-163](#) and described in [Table 13-102](#).

ePWM XBAR Mux Configuration for TRIP10

Figure 13-163. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-102. TRIP10MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

Table 13-102. TRIP10MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

Table 13-102. TRIP10MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP10 of EPWM-XBAR 00 : Select .1 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.

13.5.3.13 TRIP11MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

TRIP11MUX0TO15CFG is shown in [Figure 13-164](#) and described in [Table 13-103](#).

ePWM XBAR Mux Configuration for TRIP11

Figure 13-164. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-103. TRIP11MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

Table 13-103. TRIP11MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

Table 13-103. TRIP11MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

13.5.3.14 TRIP11MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

TRIP11MUX16TO31CFG is shown in [Figure 13-165](#) and described in [Table 13-104](#).

ePWM XBAR Mux Configuration for TRIP11

Figure 13-165. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-104. TRIP11MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

Table 13-104. TRIP11MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

Table 13-104. TRIP11MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP11 of EPWM-XBAR 00 : Select .1 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.

13.5.3.15 TRIP12MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

TRIP12MUX0TO15CFG is shown in [Figure 13-166](#) and described in [Table 13-105](#).

ePWM XBAR Mux Configuration for TRIP12

Figure 13-166. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-105. TRIP12MUX0TO15CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Group Select Bits for Mux15: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX14	R/W	0h	Group Select Bits for Mux14: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX13	R/W	0h	Group Select Bits for Mux13: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX12	R/W	0h	Group Select Bits for Mux12: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX11	R/W	0h	Group Select Bits for Mux11: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

Table 13-105. TRIP12MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX10	R/W	0h	Group Select Bits for Mux10: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX9	R/W	0h	Group Select Bits for Mux9: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX8	R/W	0h	Group Select Bits for Mux8: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX7	R/W	0h	Group Select Bits for Mux7: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX6	R/W	0h	Group Select Bits for Mux6: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX5	R/W	0h	Group Select Bits for Mux5: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX4	R/W	0h	Group Select Bits for Mux4: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

Table 13-105. TRIP12MUX0TO15CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX3	R/W	0h	Group Select Bits for Mux3: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX2	R/W	0h	Group Select Bits for Mux2: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX1	R/W	0h	Group Select Bits for Mux1: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX0	R/W	0h	Group Select Bits for Mux0: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

13.5.3.16 TRIP12MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

TRIP12MUX16TO31CFG is shown in [Figure 13-167](#) and described in [Table 13-106](#).

ePWM XBAR Mux Configuration for TRIP12

Figure 13-167. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-106. TRIP12MUX16TO31CFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Group Select Bits for Mux31: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
29-28	MUX30	R/W	0h	Group Select Bits for Mux30: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
27-26	MUX29	R/W	0h	Group Select Bits for Mux29: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
25-24	MUX28	R/W	0h	Group Select Bits for Mux28: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
23-22	MUX27	R/W	0h	Group Select Bits for Mux27: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

Table 13-106. TRIP12MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-20	MUX26	R/W	0h	Group Select Bits for Mux26: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
19-18	MUX25	R/W	0h	Group Select Bits for Mux25: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
17-16	MUX24	R/W	0h	Group Select Bits for Mux24: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
15-14	MUX23	R/W	0h	Group Select Bits for Mux23: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
13-12	MUX22	R/W	0h	Group Select Bits for Mux22: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
11-10	MUX21	R/W	0h	Group Select Bits for Mux21: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
9-8	MUX20	R/W	0h	Group Select Bits for Mux20: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

Table 13-106. TRIP12MUX16TO31CFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	MUX19	R/W	0h	Group Select Bits for Mux19: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
5-4	MUX18	R/W	0h	Group Select Bits for Mux18: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
3-2	MUX17	R/W	0h	Group Select Bits for Mux17: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.
1-0	MUX16	R/W	0h	Group Select Bits for Mux16: Selects 4X1 group output for TRIP12 of EPWM-XBAR 00 : Select .1 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.

13.5.3.17 TRIP4MUXENABLE Register (Offset = 20h) [reset = 0h]

TRIP4MUXENABLE is shown in [Figure 13-168](#) and described in [Table 13-107](#).

ePWM XBAR Mux Enable for TRIP4

Figure 13-168. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-107. 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 enabled 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.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux30 is enabled 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.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux29 is enabled 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.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux28 is enabled 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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-107. TRIP4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux26 is enabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux25 is enabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux24 is enabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-107. TRIP4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 13-107. TRIP4MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

13.5.3.18 TRIP5MUXENABLE Register (Offset = 22h) [reset = 0h]

TRIP5MUXENABLE is shown in [Figure 13-169](#) and described in [Table 13-108](#).

ePWM XBAR Mux Enable for TRIP5

Figure 13-169. TRIP5MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-108. TRIP5MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux31 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux30 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux29 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux28 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-108. TRIP5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux26 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux25 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux24 is enabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-108. TRIP5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 13-108. TRIP5MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.19 TRIP7MUXENABLE Register (Offset = 24h) [reset = 0h]

TRIP7MUXENABLE is shown in [Figure 13-170](#) and described in [Table 13-109](#).

ePWM XBAR Mux Enable for TRIP7

Figure 13-170. TRIP7MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-109. TRIP7MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux31 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux30 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux29 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux28 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-109. TRIP7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux26 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux25 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux24 is enabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-109. TRIP7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-109. TRIP7MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.20 TRIP8MUXENABLE Register (Offset = 26h) [reset = 0h]

TRIP8MUXENABLE is shown in [Figure 13-171](#) and described in [Table 13-110](#).

ePWM XBAR Mux Enable for TRIP8

Figure 13-171. TRIP8MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-110. TRIP8MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux31 is enabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux30 is enabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux29 is enabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux28 is enabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-110. TRIP8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 enabled 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.</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 enabled 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.</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 enabled 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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 13-110. TRIP8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-110. TRIP8MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.21 TRIP9MUXENABLE Register (Offset = 28h) [reset = 0h]

TRIP9MUXENABLE is shown in [Figure 13-172](#) and described in [Table 13-111](#).

ePWM XBAR Mux Enable for TRIP9

Figure 13-172. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-111. 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 enabled 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.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux30 is enabled 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.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux29 is enabled 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.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux28 is enabled 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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-111. TRIP9MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux26 is enabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux25 is enabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux24 is enabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-111. TRIP9MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 13-111. TRIP9MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.22 TRIP10MUXENABLE Register (Offset = 2Ah) [reset = 0h]

TRIP10MUXENABLE is shown in [Figure 13-173](#) and described in [Table 13-112](#).

ePWM XBAR Mux Enable for TRIP10

Figure 13-173. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-112. 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 enabled 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.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux30 is enabled 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.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux29 is enabled 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.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux28 is enabled 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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-112. TRIP10MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 enabled 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.</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 enabled 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.</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 enabled 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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 13-112. TRIP10MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-112. TRIP10MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

13.5.3.23 TRIP11MUXENABLE Register (Offset = 2Ch) [reset = 0h]

TRIP11MUXENABLE is shown in [Figure 13-174](#) and described in [Table 13-113](#).

ePWM XBAR Mux Enable for TRIP11

Figure 13-174. TRIP11MUXENABLE 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-113. 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 enabled 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.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux30 is enabled 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.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux29 is enabled 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.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux28 is enabled 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.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-113. TRIP11MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux26 is enabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux25 is enabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux24 is enabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-113. TRIP11MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-113. TRIP11MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.24 TRIP12MUXENABLE Register (Offset = 2Eh) [reset = 0h]

TRIP12MUXENABLE is shown in [Figure 13-175](#) and described in [Table 13-114](#).

ePWM XBAR Mux Enable for TRIP12

Figure 13-175. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-114. TRIP12MUXENABLE Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux31 is enabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux30 is enabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux29 is enabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux28 is enabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux27 is enabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux27 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details.

Table 13-114. TRIP12MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 enabled 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.</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 enabled 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.</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 enabled 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.</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.</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.</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.</p>
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.</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.</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.</p>

Table 13-114. TRIP12MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</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.</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.</p>
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.</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.</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.</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.</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.</p>

Table 13-114. TRIP12MUXENABLE Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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.</p>
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.</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.</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.</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.</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.</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.</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.</p>

13.5.3.25 TRIPOUTINV Register (Offset = 38h) [reset = 0h]

TRIPOUTINV is shown in [Figure 13-176](#) and described in [Table 13-115](#).

ePWM XBAR Output Inversion Register

Figure 13-176. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-115. 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.
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.
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.
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.
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.
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.

Table 13-115. TRIPOUTINV Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.
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.

13.5.3.26 TRIPLOCK Register (Offset = 3Eh) [reset = 0h]

TRIPLOCK is shown in [Figure 13-177](#) and described in [Table 13-116](#).

ePWM XBAR Configuration Lock register

Figure 13-177. TRIPLOCK Register

31	30	29	28	27	26	25	24
KEY							
R=0/W=1-0h							
23	22	21	20	19	18	17	16
KEY							
R=0/W=1-0h							
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/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-116. TRIPLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	KEY	R=0/W=1	0h	Bit-0 of this register can be set only if KEY= 0x5a5a
15-1	RESERVED	R=0	0h	Reserved
0	LOCK	R/SOnce	0h	<p>Locks the configuration for EPWM-XBAR. Once the configuration is locked, writes to the below registers for EPWM-XBAR is blocked.</p> <p>Registers Affected by the LOCK mechanism:</p> <p>EPWM-XBAROUTyMUX0TO15CFG</p> <p>EPWM-XBAROUTyMUX16TO31CFG</p> <p>EPWM-XBAROUTyMUXENABLE</p> <p>EPWM-XBAROUTLATEN</p> <p>EPWM-XBAROUTINV</p> <p>0: Writes to the above registers are allowed</p> <p>1: Writes to the above registers are blocked</p> <p>Note:</p> <p>[1] LOCK mechanism only applies to writes. Reads are never blocked.</p>

13.5.4 TRIG_REGS Registers

Table 13-117 lists the memory-mapped registers for the TRIG_REGS. All register offset addresses not listed in Table 13-117 should be considered as reserved locations and the register contents should not be modified.

Table 13-117. TRIG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SYNCSELECT	Sync Input and Output Select Register	EALLOW	Go
2h	EXTADCSOCSELECT	External ADCSOC Select Register	EALLOW	Go
4h	SYNCSOCLOCK	SYNCSEL and EXTADCSOC Select Lock register	EALLOW	Go

13.5.4.1 SYNCSELECT Register (Offset = 0h) [reset = 0h]

SYNCSELECT is shown in [Figure 13-178](#) and described in [Table 13-118](#).

Sync Input and Output Select Register

Figure 13-178. SYNCSELECT Register

31	30	29	28	27	26	25	24
RESERVED			SYNCOUT			RESERVED	
R=0-0h			R/W-0h			R=0-0h	
23	22	21	20	19	18	17	16
RESERVED							
R=0-0h							
15	14	13	12	11	10	9	8
RESERVED	ECAP4SYNCIN			ECAP1SYNCIN			EPWM10SYNCIN
R=0-0h		R/W-0h			R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
EPWM10SYNCIN		EPWM7SYNCIN			EPWM4SYNCIN		
R/W-0h		R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-118. SYNCSELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-29	RESERVED	R=0	0h	Reserved
28-27	SYNCOUT	R/W	0h	Select Syncout Source: 00: EPWM1SYNCOUT selected 01: EPWM4SYNCOUT selected 10: EPPW7SYNCOUT selected 11: EPWM10SYNCOUT selected
26-16	RESERVED	R=0	0h	Reserved
15	RESERVED	R=0	0h	Reserved
14-12	ECAP4SYNCIN	R/W	0h	Selects Sync Input Source for ECAP4: 000: EPWM1SYNCOUT selected 001: EPWM4SYNCOUT selected 010: EPPW7SYNCOUT selected 011: EPWM10SYNCOUT selected 100: ECAP1SYNCOUT selected 101: EXTSYNCIN1 selected 110: EXTSYNCIN2 selected 111: Reserved Notes: [1] Reserved position defaults to 000 selection

Table 13-118. SYNCSELECT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-9	ECAP1SYNCIN	R/W	0h	<p>Selects Sync Input Source for ECAP1:</p> <p>000: EPWM1SYNCOOUT selected</p> <p>001: EPWM4SYNCOOUT selected</p> <p>010: EPPW7SYNCOOUT selected</p> <p>011: EPWM10SYNCOOUT selected</p> <p>100: ECAP1SYNCOOUT selected (Reserved)</p> <p>101: EXTSYNCIN1 selected</p> <p>110: EXTSYNCIN2 selected</p> <p>111: Reserved</p> <p>Notes:</p> <p>[1] Reserved position defaults to 000 selection</p>
8-6	EPWM10SYNCIN	R/W	0h	<p>Selects Sync Input Source for EPWM10:</p> <p>000: EPWM1SYNCOOUT selected</p> <p>001: EPWM4SYNCOOUT selected</p> <p>010: EPPW7SYNCOOUT selected</p> <p>011: EPWM10SYNCOOUT selected (Reserved)</p> <p>100: ECAP1SYNCOOUT selected (Reserved)</p> <p>101: EXTSYNCIN1 selected</p> <p>110: EXTSYNCIN2 selected</p> <p>111: Reserved</p> <p>Notes:</p> <p>[1] Reserved position defaults to 000 selection</p>
5-3	EPWM7SYNCIN	R/W	0h	<p>Selects Sync Input Source for EPWM7:</p> <p>000: EPWM1SYNCOOUT selected</p> <p>001: EPWM4SYNCOOUT selected</p> <p>010: EPPW7SYNCOOUT selected (Reserved)</p> <p>011: EPWM10SYNCOOUT selected (Reserved)</p> <p>100: ECAP1SYNCOOUT selected (Reserved)</p> <p>101: EXTSYNCIN1 selected</p> <p>110: EXTSYNCIN2 selected</p> <p>111: Reserved</p> <p>Notes:</p> <p>[1] Reserved position defaults to 000 selection</p>
2-0	EPWM4SYNCIN	R/W	0h	<p>Selects Sync Input Source for EPWM4:</p> <p>000: EPWM1SYNCOOUT selected</p> <p>001: EPWM4SYNCOOUT selected (Reserved)</p> <p>010: EPPW7SYNCOOUT selected (Reserved)</p> <p>011: EPWM10SYNCOOUT selected (Reserved)</p> <p>100: ECAP1SYNCOOUT selected (Reserved)</p> <p>101: EXTSYNCIN1 selected</p> <p>110: EXTSYNCIN2 selected</p> <p>111: Reserved</p> <p>Notes:</p> <p>[1] Reserved position defaults to 000 selection</p>

13.5.4.2 EXTADCSOCSELECT Register (Offset = 2h) [reset = 0h]

EXTADCSOCSELECT is shown in [Figure 13-179](#) and described in [Table 13-119](#).

External ADCSOC Select Register

Figure 13-179. EXTADCSOCSELECT Register

31	30	29	28	27	26	25	24
RESERVED				PWM12SOCBE N	PWM11SOCBE N	PWM10SOCBE N	PWM9SOCBE N
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-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				PWM12SOCAE N	PWM11SOCAE N	PWM10SOCAE N	PWM9SOCAE N
R=0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-119. EXTADCSOCSELECT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R=0	0h	Reserved
27	PWM12SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
26	PWM11SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
25	PWM10SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
24	PWM9SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
23	PWM8SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
22	PWM7SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
21	PWM6SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
20	PWM5SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected

Table 13-119. EXTADCSOCSELECT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	PWM4SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
18	PWM3SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
17	PWM2SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
16	PWM1SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected
15-12	RESERVED	R=0	0h	Reserved
11	PWM12SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
10	PWM11SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
9	PWM10SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
8	PWM9SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
7	PWM8SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
6	PWM7SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
5	PWM6SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
4	PWM5SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
3	PWM4SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
2	PWM3SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected
1	PWM2SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected

Table 13-119. EXTADCSOCSELECT Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	PWM1SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected

13.5.4.3 SYNCLOCK Register (Offset = 4h) [reset = 0h]

SYNCLOCK is shown in [Figure 13-180](#) and described in [Table 13-120](#).

SYNCSEL and EXTADCSOC Select Lock register

Figure 13-180. 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						EXTADCSOC ELECT	SYNCSELECT
R=0-0h						R/SOnce-0h	R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 13-120. SYNCLOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-2	RESERVED	R=0	0h	Reserved
1	EXTADCSOCSELECT	R/SOnce	0h	EXTADCSOCSELECT 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 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
0	SYNCSELECT	R/SOnce	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 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

High-Resolution Pulse Width Modulator (HRPWM)

This document is used in conjunction with the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*. The HRPWM module described in this reference guide is a Type 2 HRPWM. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* ([SPRU566](#)) for a list of all devices with an HRPWM module of the same type, to determine the differences between types, and for a list of device-specific differences within a type.

Topic	Page
14.1 Introduction	1759
14.2 Operational Description of HRPWM.....	1761
14.3 Appendix A: SFO Library Software - SFO_TI_Build_V7.lib	1782

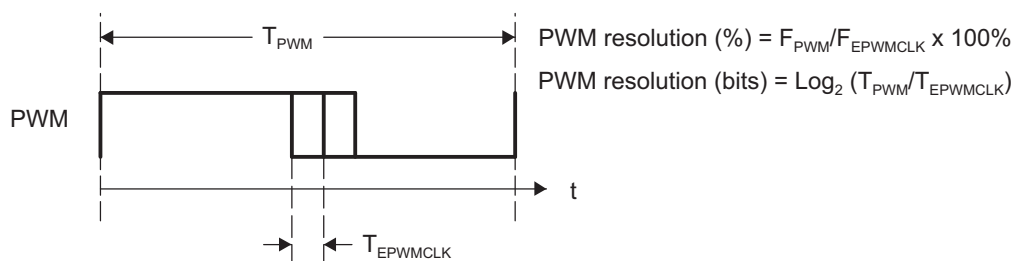
14.1 Introduction

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 14-1](#), the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

Figure 14-1. 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, [Table 14-1](#) 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 14-1. Resolution for PWM and HRPWM

PWM Freq (kHz)	Regular Resolution (PWM)		High Resolution (HRPWM)	
	100 MHz EPWMCLK			
	Bits	%	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:

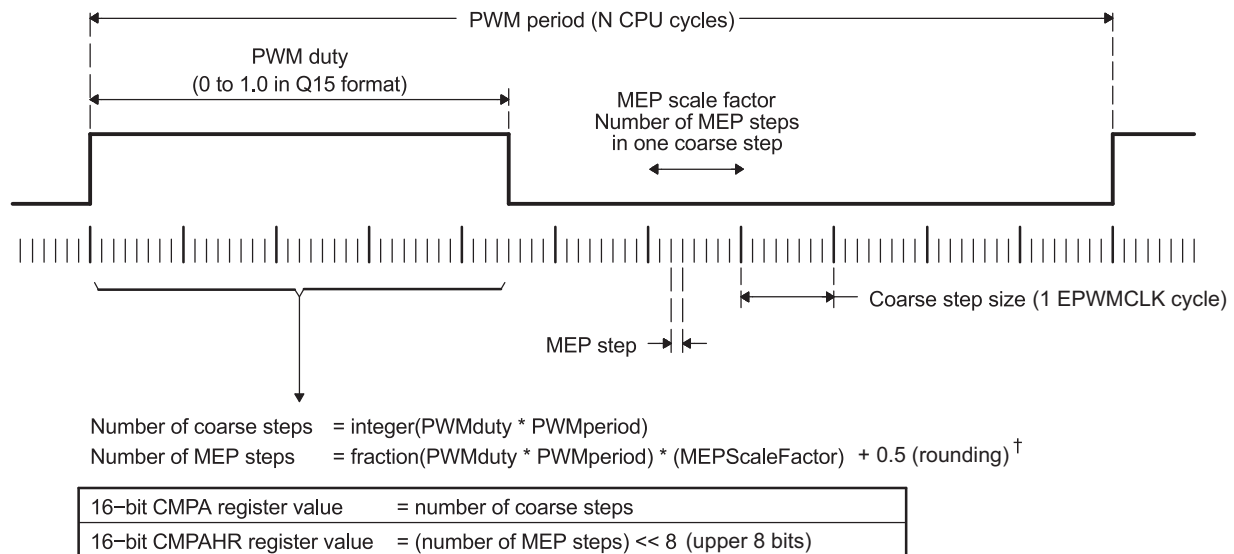
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

14.2 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 14.2.6](#).

[Figure 14-2](#) 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 14-2. Operating Logic Using MEP



[†] For MEP range and rounding adjustment. (0x0080 in Q8 format)

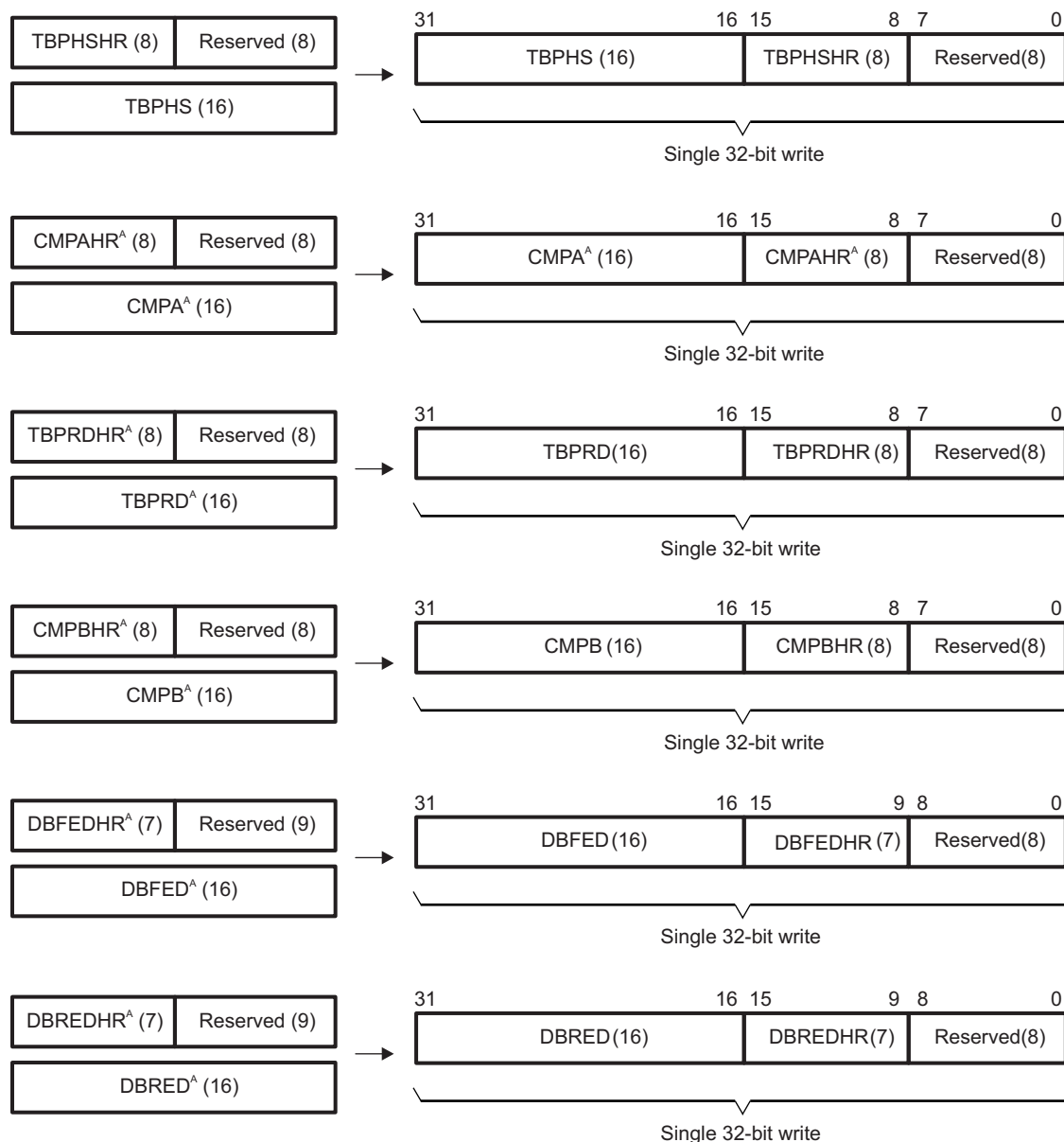
To generate an HRPWM waveform, configure the ePWM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the ePWM registers to extend edge resolution. Although many programming combinations are possible, only a few are needed and practical. These methods are described in [Section 14.2.7](#).

Registers discussed but not found in this document can be seen in the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*.

14.2.1 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
- TBPRDHR - Time Base Period High Resolution Register. (available on some devices)
- CMPBHR - Compare B High Resolution Register
- DBREDHR - Deadband Generator Rising Edge Delay High Resolution Register
- DBFEDHR - Deadband Generator Falling Edge Delay High Resolution Register

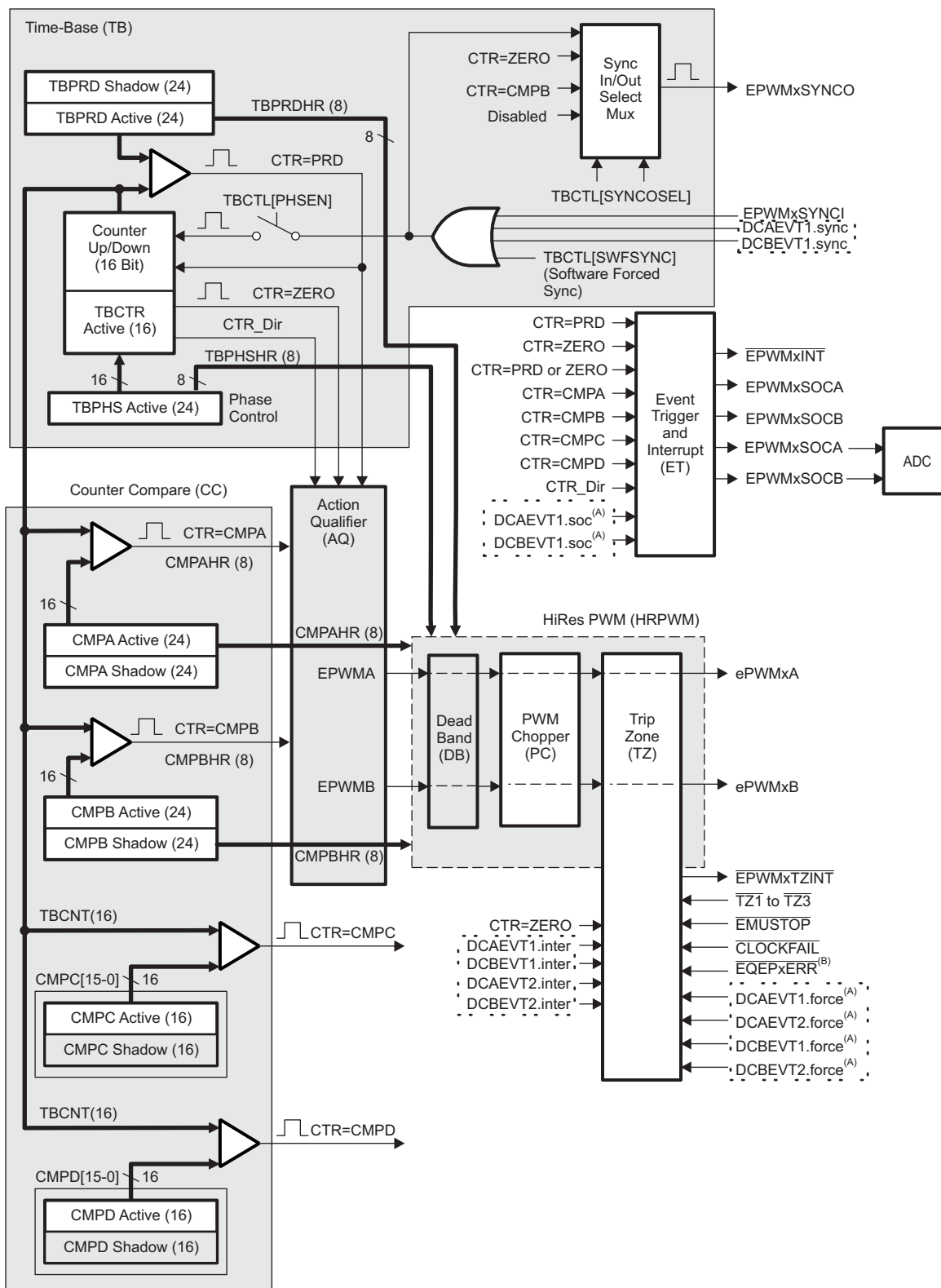
Figure 14-3. HRPWM Extension Registers and Memory Configuration


- 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.

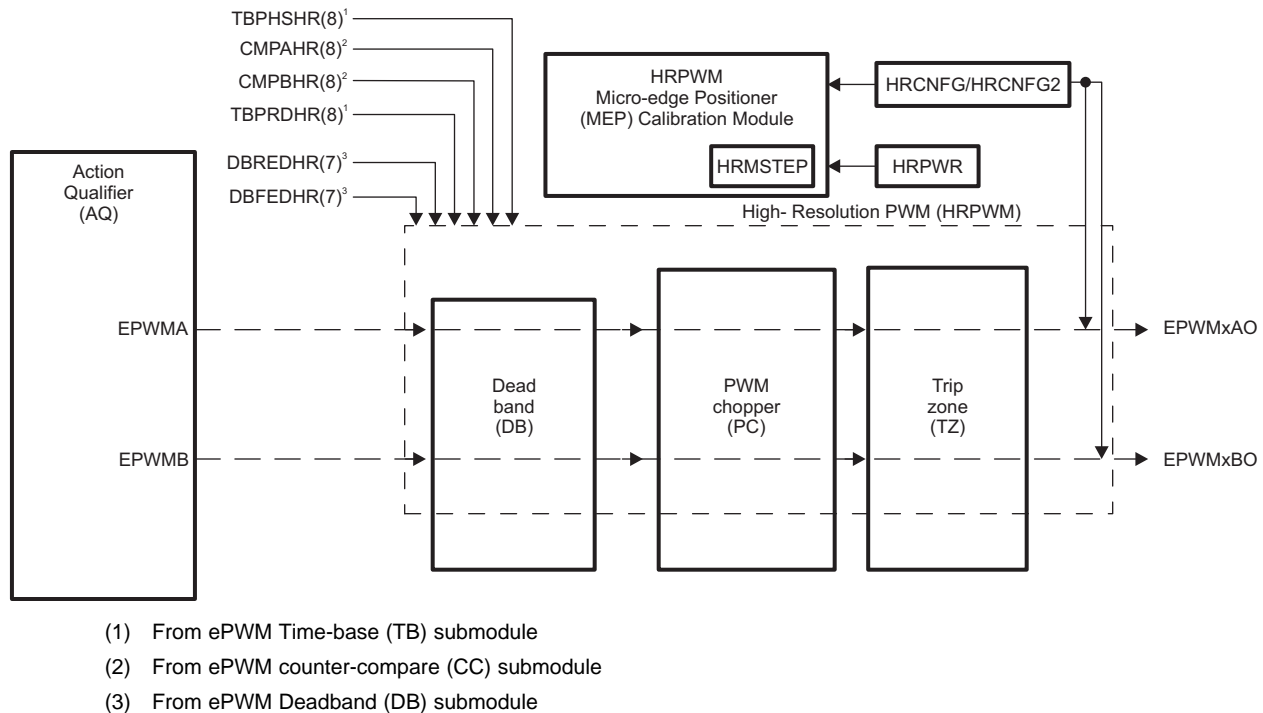
NOTE: HRPWM capabilities on Deadband Rising Edge Delay and Falling Edge Delay is applicable only during Dead Band half cycle clocking Operation. The number of MEP steps is half in size [bits 15:9]than duty and phase high resolution registers for the same reason

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. [Figure 14-4](#) shows how the HRPWM interfaces with the 8-bit extension registers.

Figure 14-4. HRPWM System Interface



A These events are generated by the ePWM digital compare (DC) submodule.

Figure 14-5. HRPWM Block Diagram


14.2.2 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 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 2 HRPWM module, below is a description of the Auto-conversion Mode taking CMPAHR as an example. If auto-conversion is enabled, $CMPAHR = \text{fraction}(PWMduty * PWMperiod) < 8$. 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 $CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) < 8$. 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. $CMPBHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) < 8$.

14.2.3 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.

14.2.4 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 14-2](#) shows the typical range of operating frequencies supported by the HRPWM.

Table 14-2. Relationship Between MEP Steps, PWM Frequency and Resolution

System (MHz)	MEP Steps Per EPWMCLK ⁽¹⁾⁽²⁾⁽³⁾	PWM MIN (Hz) ⁽⁴⁾	PWM MAX (MHz)	Res. @ MAX (Bits) ⁽⁵⁾
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.

14.2.4.1 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 14-6, 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 14-3.

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. Table 14-3 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.

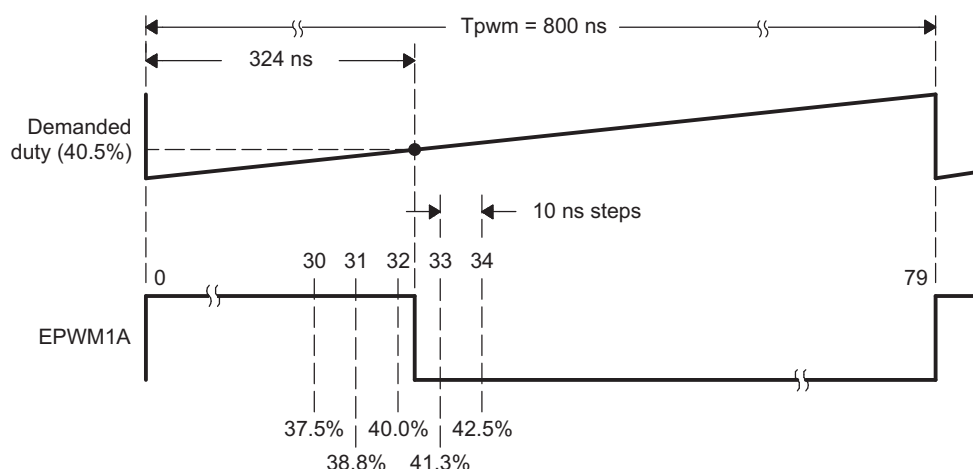
Figure 14-6. Required PWM Waveform for a Requested Duty = 40.5%


Table 14-3. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)

CMPA (count)^{(1) (2) (3)}	DUTY %	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	35.0	280	32	18	40.405	323.24
29	36.3	290	32	19	40.428	323.42
30	37.5	300	32	20	40.450	323.60
31	38.8	310	32	21	40.473	323.78
32	40.0	320	32	22	40.495	323.96
33	41.3	330	32	23	40.518	324.14
34	42.5	340	32	24	40.540	324.32
			32	25	40.563	324.50
Required			32	26	40.585	324.68
32.40	40.5	324	32	27	40.608	324.86

⁽¹⁾ TBCLK = 100 MHz, 10 ns

⁽²⁾ 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

⁽³⁾ Assumed MEP step size for the above example = 180 ps
See the device-specific data manual for typical and maximum MEP values.

14.2.4.2 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, PWMDuty	= 0.405 (40.5%)
PWM period in terms of coarse steps, PWMPeriod (800 ns/10 ns)	= 80
Number of MEP steps per coarse step at 180 ps (10 n/180 ps), MEP_ScaleFactor	= 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	= int(PWMDuty * PWMPeriod); int means integer part
	= int(0.405 * 80)
	= int(32.4)
CMPA register value	= 32 (20h)

Step 2: Fractional value conversion for CMPAHR register

CMPAHR	= (frac(PWMDuty * PWMPeriod)* MEP_ScaleFactor + 0.5) << 8); frac means fractional part
	= (frac(32.4) * 55 + 0.5) << 8; Shifting is to move the value to the high byte of CMPAHR.
	= (0.4 * 55 + 0.5) << 8
	= (22 + 0.5) << 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 = frac (PWMDuty*PWMperiod<<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 14.2.7](#). The actual implementation takes advantage of the 32-bit CPU architecture of the 28xx, and is somewhat different from the steps shown in [Section 14.2.4.2](#).

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.

14.2.4.3 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 3.

Duty cycle range limitations are illustrated in [Figure 14-7](#) to [Figure 14-10](#). 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 14-4](#). 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 14-7. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)

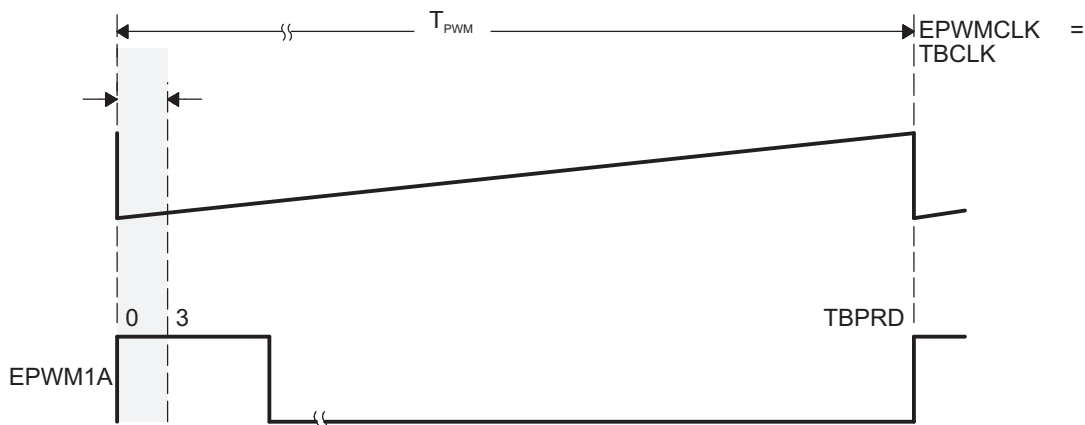


Table 14-4. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles

PWM Frequency ⁽¹⁾ (kHz)	3 Cycles Minimum Duty	3 Cycles Maximum Duty ⁽²⁾
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%

⁽¹⁾ EPWMCLK = TBCLK = 100 MHz

⁽²⁾ 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 14-8](#). 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 14-4](#).

Figure 14-8. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)

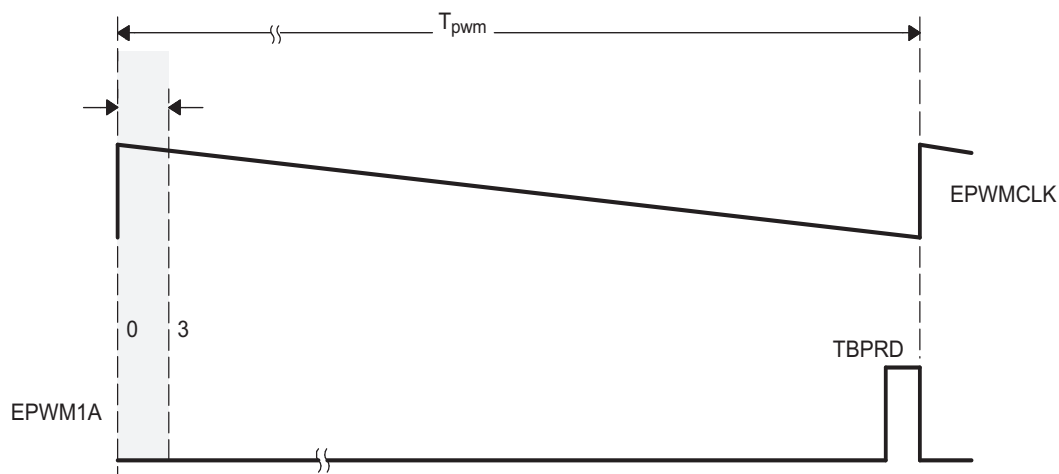


Figure 14-9. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)

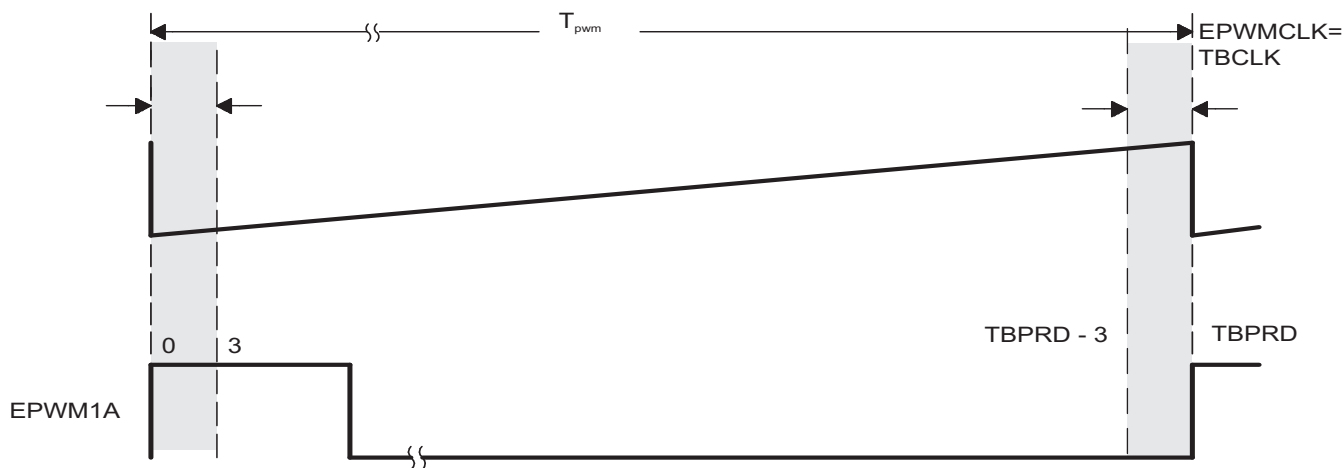
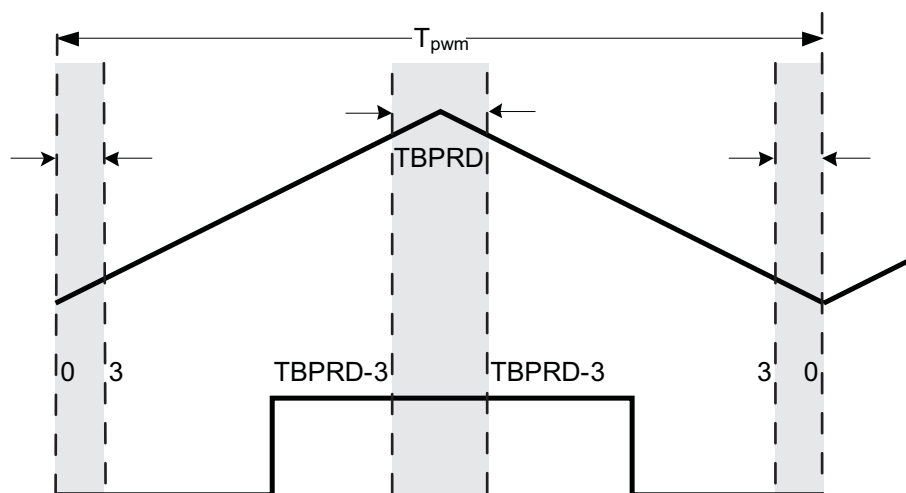


Figure 14-10. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)


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.

14.2.4.4 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 14.2.4.2](#) 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

Integer period value	$= 571 * T_{TBCLK}$ $= \text{int}(571.428) * T_{TBCLK}$ $= \text{int}(\text{PWMperiod}) * T_{TBCLK}$
In up-count mode: TBPRD	$= 570$ (TBPRD = period value - 1) $= 023Ah$
In up-down count mode: TBPRD	$= 285$ (TBPRD = period value / 2) $= 011Dh$

Step 2: Fractional value conversion for TBPRDHR register

TBPRDHR register value	$= (\text{frac}(\text{PWMperiod}) * \text{MEP_ScaleFactor} + 0.5)$
If auto-conversion enabled and HRMSTEP = MEP_ScaleFactor value (55):	$= \text{frac}(\text{PWMperiod}) \ll 8$ (Shifting is to move the value to the high byte of TBPRDHR)
TBPRDHR register value	$= \text{frac}(571.428) \ll 8$ $= 0.428 \times 256$ $= 6D00h$

The autoconversion will then automatically perform the calculation such that TBPRDHR MEP delay is scaled by hardware to:

$$=((\text{TBPRDHR}(15:0) \gg 8) \times \text{HRMSTEP} + 80h) \ll 8$$

$$= (006Dh \times 55 + 80h) \gg 8$$

$$=(17EBh) \gg 8$$

Period MEP delay

$$=0017h \text{ MEP Steps}$$

14.2.4.4.1 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 14.3](#).
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.

14.2.5 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.

14.2.6 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_V7.lib software can be found in [Section 14.3](#).

14.2.7 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 14-1](#) This example assumes MEP step size of 150 ps and does not use the SFO library.

Example 14-1. #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
```

14.2.7.1 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 14-11 and Figure 14-12 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 14-11. Simple Buck Controlled Converter Using a Single PWM

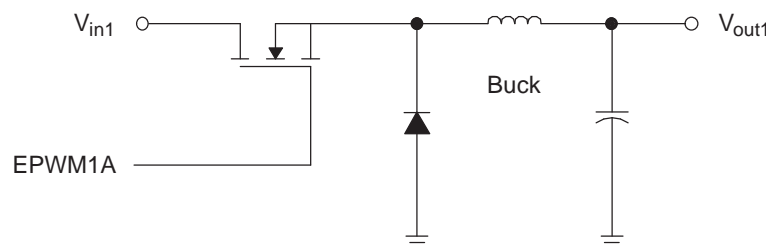
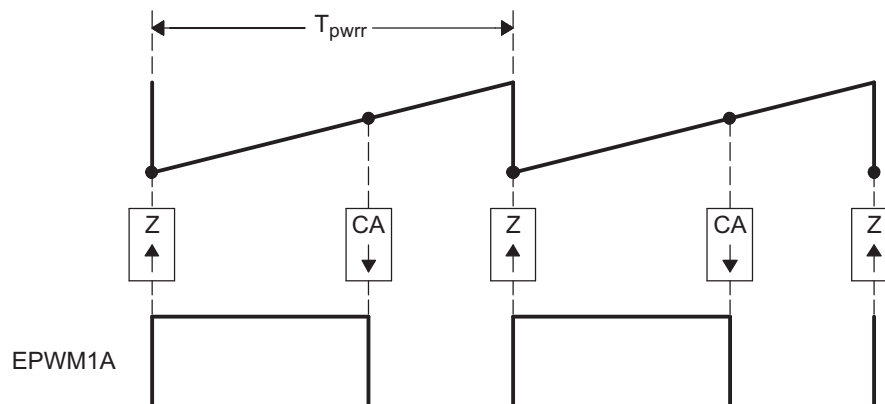


Figure 14-12. PWM Waveform Generated for Simple Buck Controlled Converter



The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

[Example 14-2](#) 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 14-2. 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.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
                                                    // Note: Use SFO functions to update
                                                    // MEP_ScaleFactor dynamically
}
```

[Example 14-3](#) shows an assembly example of run-time code for the HRPWM buck converter.

Example 14-3. 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

```

14.2.7.2 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 14-13 and Figure 14-14 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 14-13. Simple Reconstruction Filter for a PWM-based DAC

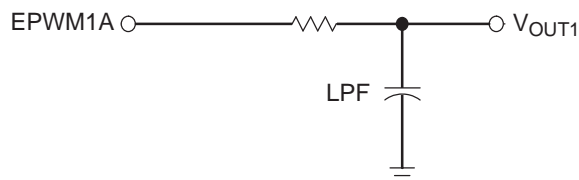
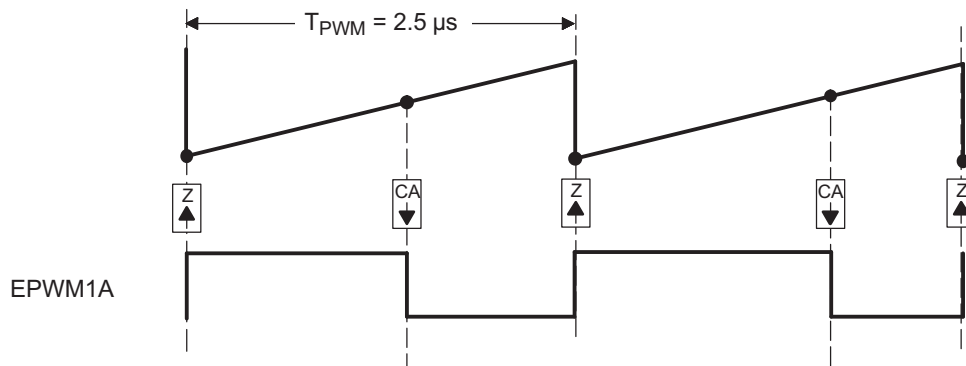


Figure 14-14. PWM Waveform Generated for the PWM DAC Function



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 14-4](#) shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

Example 14-4. 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 14-5](#) shows an assembly example of run-time code that can execute in a high-speed ISR loop.

Example 14-5. 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

```

14.3 Appendix A: SFO Library Software - SFO_TI_Build_V7.lib

The following table lists several features of the SFO_TI_Build_V7.lib library.

Table 14-5. SFO Library Features

	SFO_TI_Build_V7.lib	Unit
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.

14.3.1 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:

MEP_ScaleFactor = 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 14.2.4.3](#)).
- 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:
 - CMPAHR = (fraction(PWMduty * PWMperiod) * 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.

14.3.2 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 14-6](#).

Table 14-6. Factor Values

Software Function call	Functional Description	Updated Variables
SFO()	Returns MEP scale factor in the HRMSTEP register	MEP_ScaleFactor & 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_V7.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 14-6. A Sample of How to Add "Include" Files

```
#include "F28x7x_Device.h"           // F28x7x Headerfile
#include "F28x7x_EPwm_defines.h" // init defines
#include "SFO_V7.h"                  // SFO lib functions (needed for HRPWM)
```

Step 2. Element Declaration

Declare an integer variable for the scale factor value as shown below.

Example 14-7. 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 14-8. 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 14-9. 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)
}
```

Enhanced Capture (eCAP)

This chapter describes the enhanced capture (eCAP), which is used in systems where accurate timing of external events is important.

The eCAP module described in this reference guide is a Type 0 eCAP. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide* ([SPRU566](#)) for a list of all devices with a eCAP module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Topic	Page
15.1 Introduction	1786
15.2 Description	1786
15.3 Configuring Device Pins for the eCAP	1786
15.4 Capture and APWM Operating Mode	1787
15.5 Capture Mode Description	1788
15.6 Application of the ECAP Module	1796
15.7 Application of the APWM Mode	1805
15.8 Registers	1807

15.1 Introduction

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
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

15.2 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:

- Dedicated input capture pin
- 32-bit time base (counter)
- 4 x 32-bit time-stamp capture registers (CAP1-CAP4)
- 4-stage sequencer (Modulo4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (CAP1-CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

15.3 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 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. Since the GPIO mode is used the GPyINV register can invert the signals. .

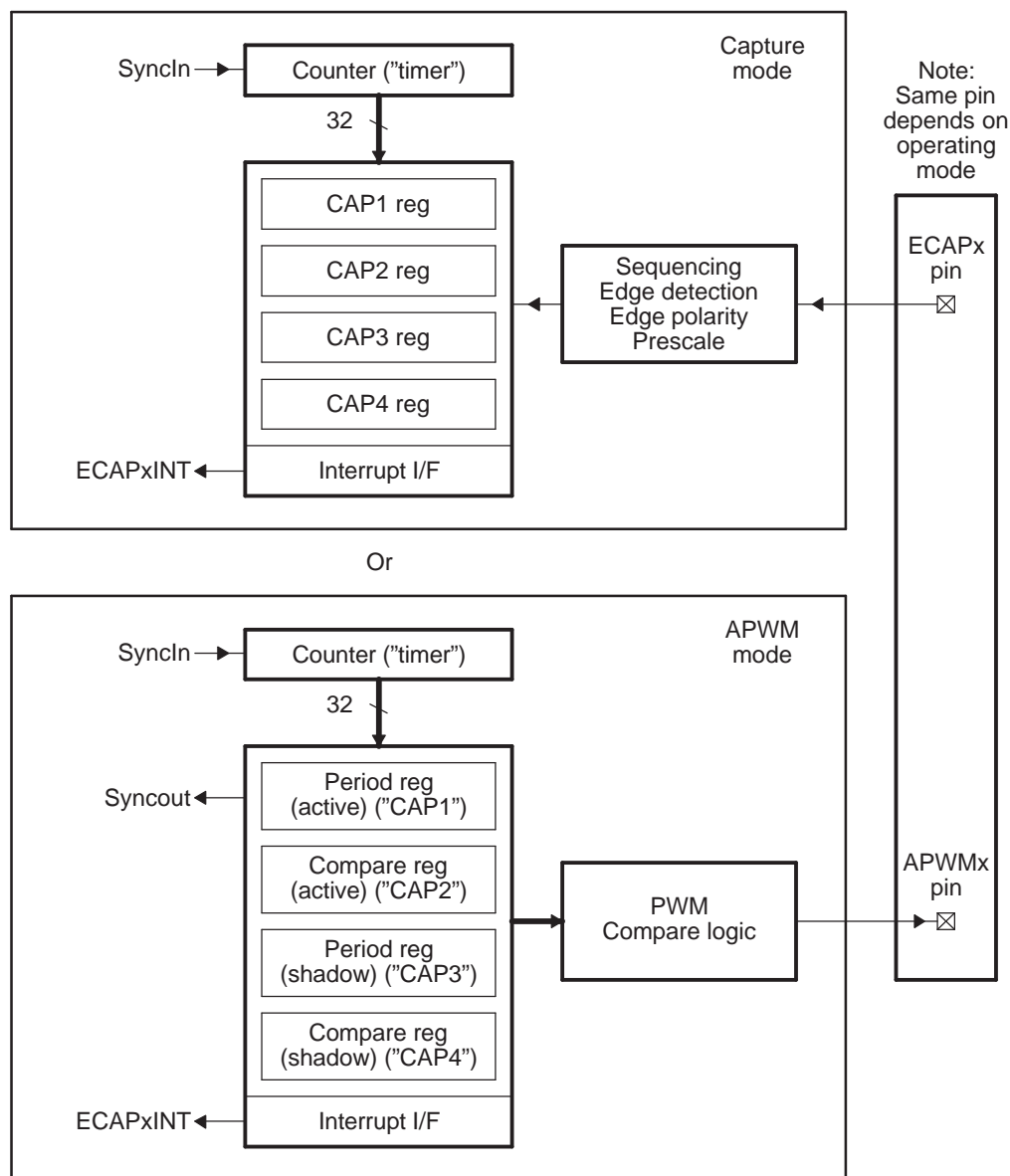
The Output X-BAR must be used to connect output signals to the OUTPUTXBARN output locations. The GPIO mux then be configured to connect the OUTPUTXBARN lines to any of several IO pins with the GPIO Mux. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding 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.

15.4 Capture and APWM Operating Mode

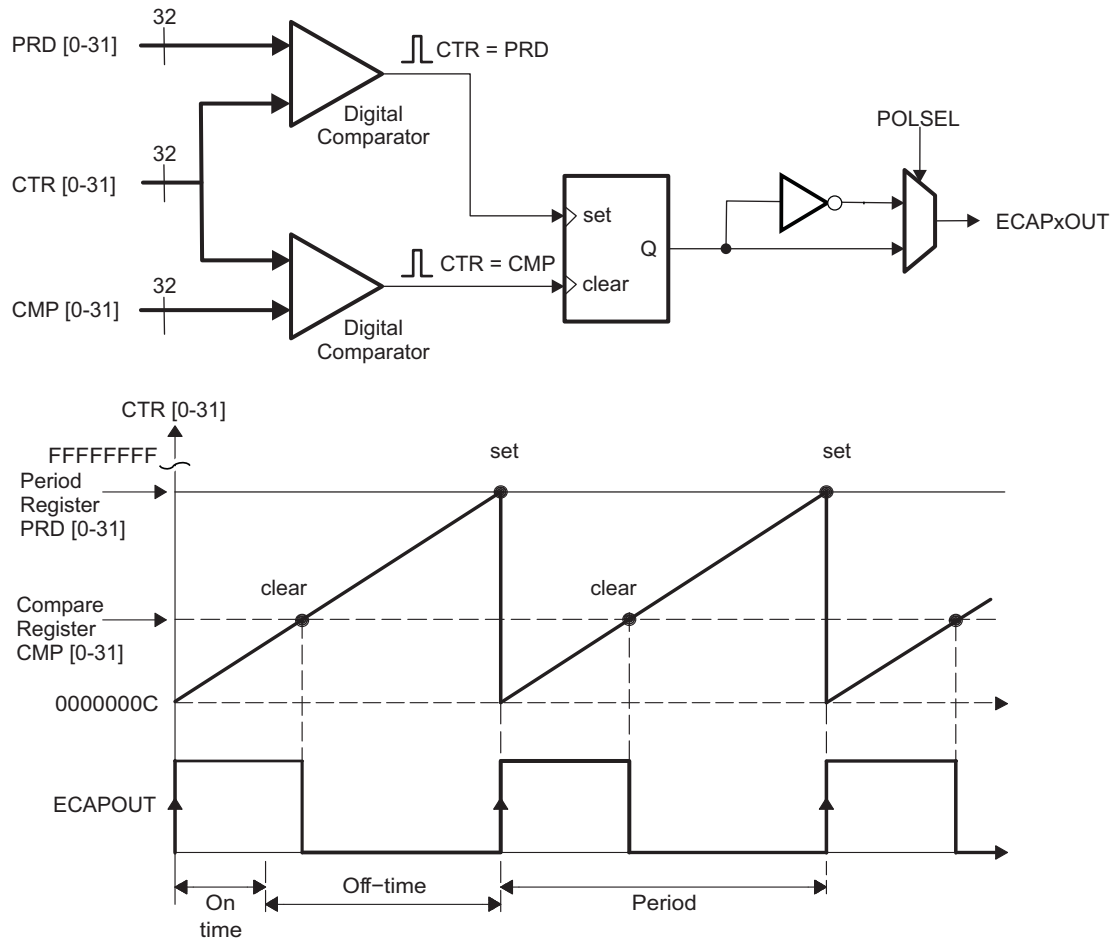
You can use the eCAP module resources to implement a single-channel PWM generator (with 32-bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 15-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 15-1. 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 15-2 further describes the output of the eCAP in APWM mode based on the CMP and PRD values.

Figure 15-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode


15.5 Capture Mode Description

Figure 15-3 shows the various components that implement the capture function.

The diagram illustrates the internal architecture of the eCAP module, organized into several functional blocks and their interconnections:

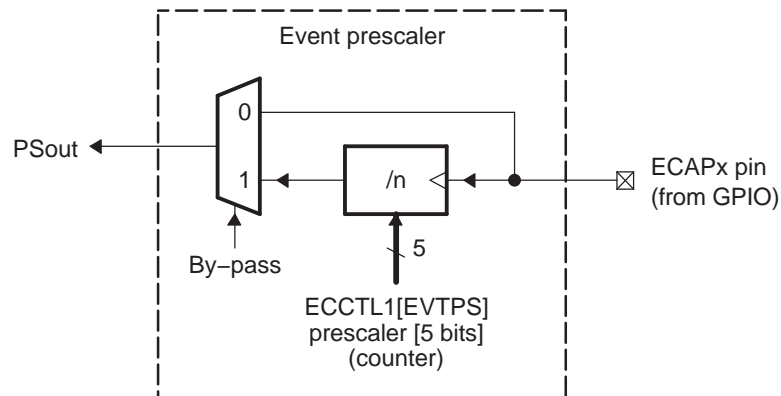
- ECCTL2 [SYNCL_EN, SYNCOSEL, SWSYNCR]:** This block contains the **CTRPHS** (phase register-32 bit) and **TSCTR** (counter-32 bit). It receives **SYNCLn** and **SYNCRn** inputs. The **TSCTR** outputs **CTR_OVF** and **RST** (Delta-mode). A 32-bit bus connects ECCTL2 to the **CTR [0-31]** and **PRD [0-31]** registers.
- ECCTL2[CAP/APWM]:** This block is divided into two modes:
 - APWM mode:** Contains **CTR [0-31]**, **PRD [0-31]**, and **CMP [0-31]** registers. These feed into **PWM compare logic**, which outputs **CTR=PRD** and **CTR=CMP** signals.
 - APWM mode (dashed box):** This section is currently inactive.
- ECCTL1 [CAPLDEN, CTRRSTx]:** This block contains the **Event qualifier** and four **Polarity select** blocks. It receives **LD1** through **LD4** signals from the capture comparators. It also receives **ECCTL1[EVTPS]** and **ECCTL1[CAPxPOL]** signals. The **Event qualifier** outputs **Capture events** (4-bit) to the **Continuous / Oneshot Capture Control** block.
- Capture Comparators:** Four comparators are shown:
 - CAP1 (APRD active):** Receives **CTR [0-31]** and **PRD [0-31]**. It outputs **APRD shadow** (32-bit) to **CAP2** and **CMP [0-31]** to the **Event qualifier**.
 - CAP2 (ACMP active):** Receives **APRD shadow** and **CMP [0-31]**. It outputs **ACMP shadow** (32-bit) to **CAP3** and **LD** to the **Event qualifier**.
 - CAP3 (APRD shadow):** Receives **ACMP shadow** and outputs **LD** to the **Event qualifier**.
 - CAP4 (ACMP shadow):** Receives **LD** from **CAP3** and outputs **LD** to the **Event qualifier**.
- Continuous / Oneshot Capture Control:** Receives **Capture events** (4-bit) and **CEVT[1:4]** signals. It outputs **CTR_OVF**, **CTR=PRD**, and **CTR=CMP** signals to the **Interrupt Trigger and Flag control** block.
- Interrupt Trigger and Flag control:** Receives **CTR_OVF**, **CTR=PRD**, and **CTR=CMP** signals. It outputs **to PIE** (Peripheral Interrupt Enable) signals.
- MODE SELECT:** A vertical block that receives **ECAPx** (mode select input) and **ECCTL1[EVTPS]** signals. It outputs **ECCTL1[CAPxPOL]** signals to the **Polarity select** blocks.

Registers: ECEINT, ECFLG, ECCLR, ECFRC

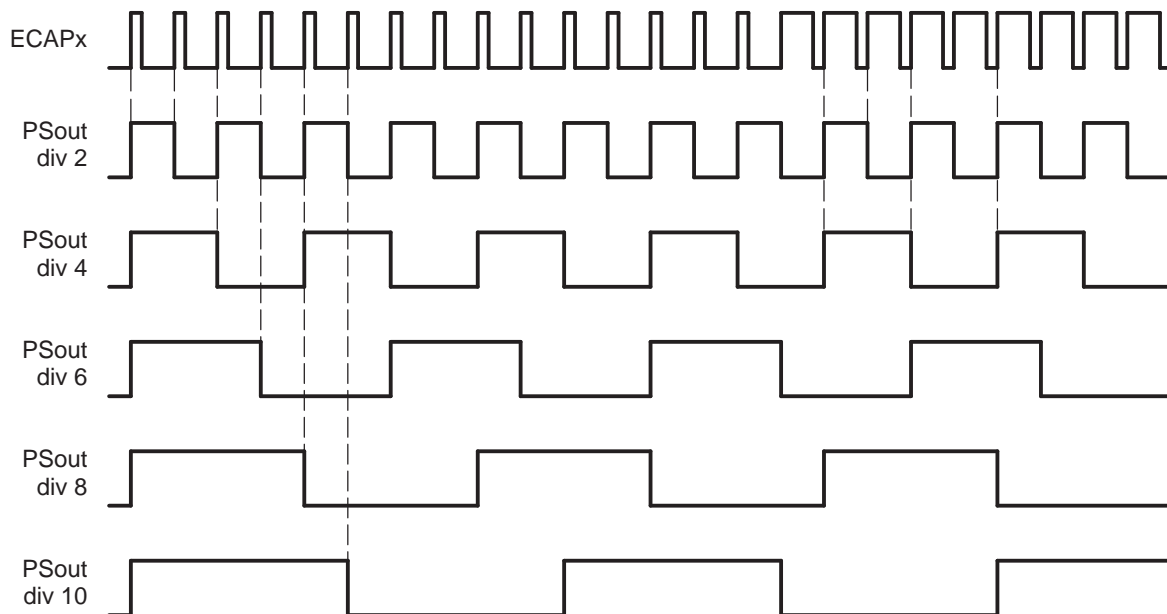
ECCTL2 [RE-ARM, CONT/ONESHT, STOP_WRAP]

- 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 15-4](#) shows a functional diagram and [Figure 15-5](#) shows the operation of the prescale function.

Figure 15-4. Event Prescale Control


- A When a prescale value of 1 is chosen (ECCTL1[13:9] = 0,0,0,0,0) the input capture signal by-passes the prescale logic completely.

Figure 15-5. Prescale Function Waveforms


15.5.2 Edge Polarity Select and Qualifier

- 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.

15.5.3 Continuous/One-Shot Control

- 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.

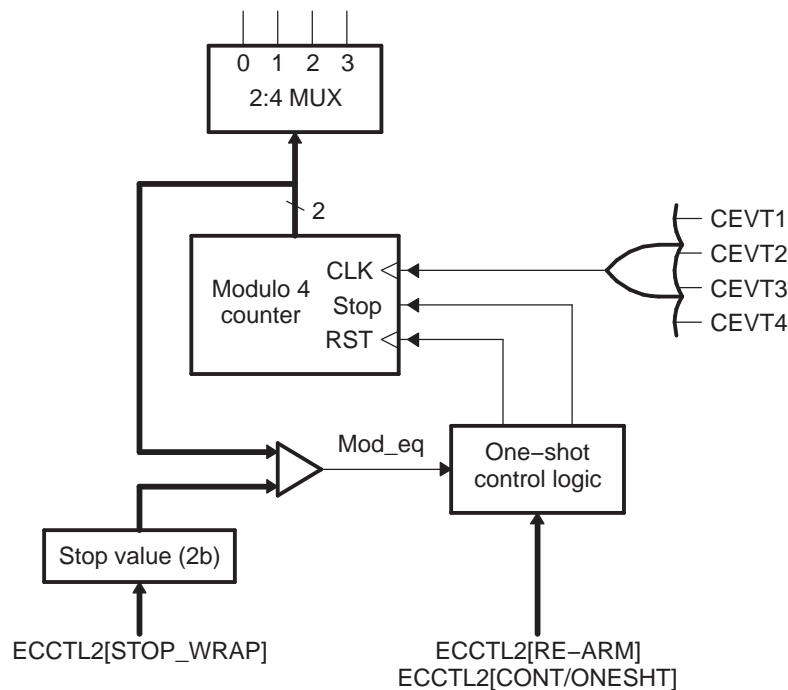
The continuous/one-shot block controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

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 15-6. Details of the Continuous/One-shot Block

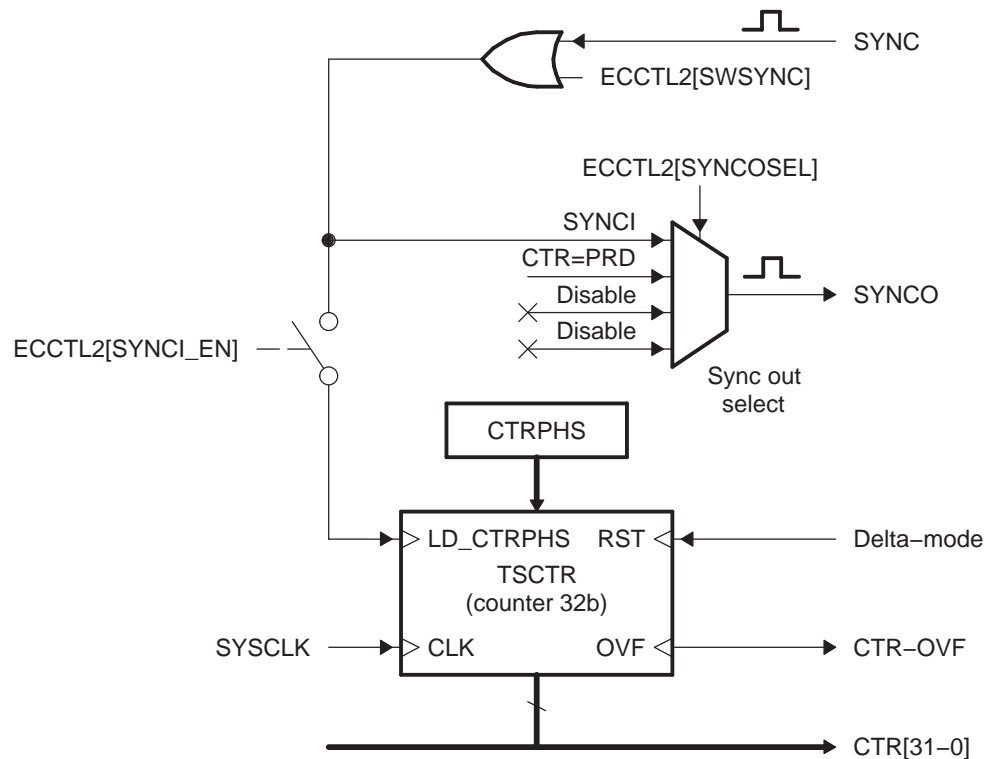


15.5.4 32-Bit Counter and Phase Control

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 15-7. Details of the Counter and Synchronization Block


15.5.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via control bit CAPLDEN. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

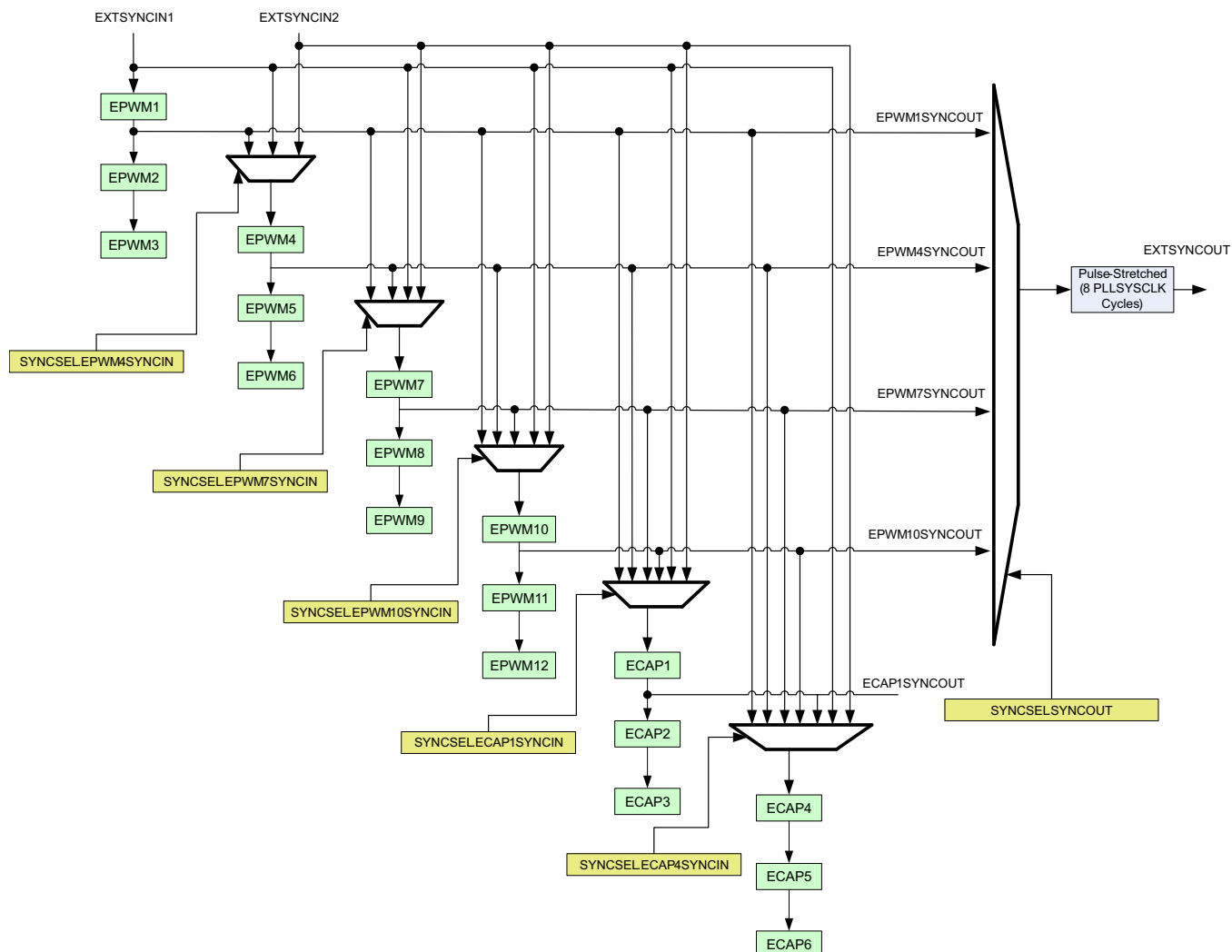
CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

15.5.6 Using SWSYNC with the ECAP Module

The SWSYNC of the ECAP module is logical OR'd with the SYNC signal as shown in [Figure 15-7](#). The SYNC signal is defined by the selection in the SYNCSEL[ECAPxSYNIN] bit for ECAP1 and ECAP4 as shown in [Figure 15-8](#).

Figure 15-8. Time-Base Counter Synchronization Scheme 4



Other ECAP modules receive the SYNC signal from the previous ECAP module. To use SWYNC with ECAP1 and ECAP4, the following workaround can be implemented:

- 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[ECAPxSYNCIN] = 0x101. This will take ECAPx.EXTSYNCIN to an inactive state.

To use SWSYNC with other ECAP modules, take measures to ensure that the previous ECAP chain is not generating a SYNCOUT signal which will interfere with the software synchronization.

15.5.7 Interrupt Control

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.

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.

15.5.8 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:

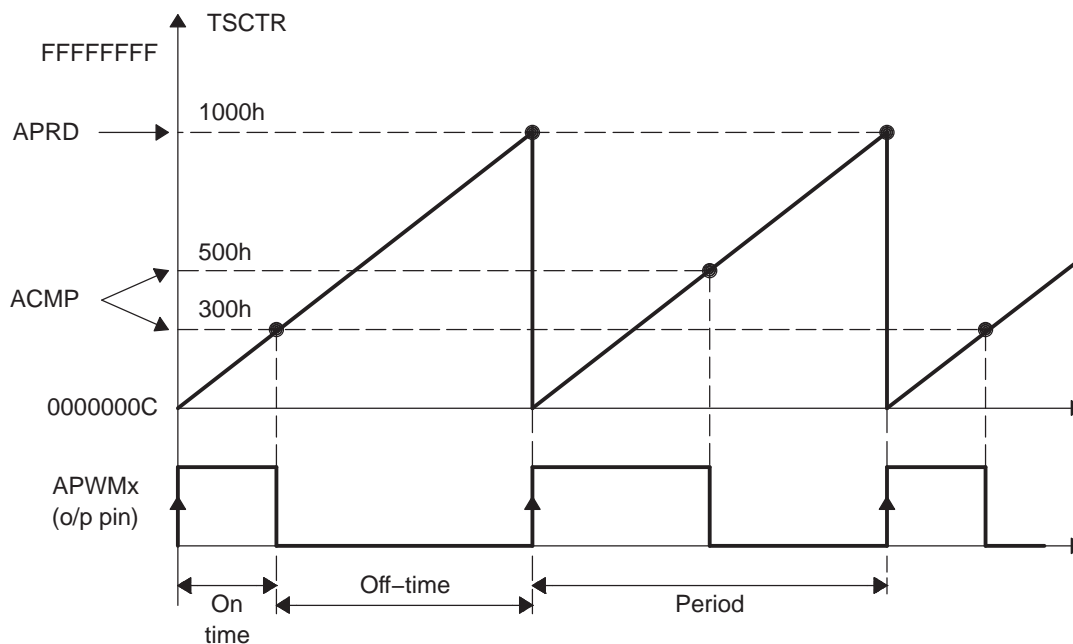
- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, $CTR[31:0] = PRD[31:0]$

15.5.9 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 15-10. 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

```

15.6 Application of the ECAP Module

The following sections will provide Applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, the examples use the eCAP “C” header files. Below are useful #defines which will help in the understanding of the examples.

```

// ECCTL1 (ECAP Control Reg 1)

//=====

// CAPxPOL bits      #define EC_RISING 0x0      #define EC_FALLING 0x1
// CTRRSTx bits      #define EC_ABS_MODE 0x0    #define EC_DELTA_MODE 0x1
// PRESCALE bits      #define EC_BYPASS 0x0      #define EC_DIV1 0x0 #define EC_DIV2 0x1
                                                #define EC_DIV4 0x2 #define EC_DIV6 0x3
                                                #define EC_DIV8 0x4 #define EC_DIV10 0x5

// ECCTL2 ( ECAP Control Reg 2)
//=====
// CONT/ONESHOT bit  #define EC_CONTINUOUS 0x0  #define EC_ONESHOT 0x1
// STOPVALUE bit     #define EC_EVENT1 0x0     #define EC_EVENT2 0x1 #define EC_EVENT3 0x2
                                                #define EC_EVENT4 0x3

// RE-ARM bit #define EC_ARM 0x1
// TSCTRSTOP bit    #define EC_FREEZE 0x0      #define EC_RUN 0x1
// SYNCO_SEL bit    #define EC_SYNCIN 0x0      #define EC_CTR_PRD 0x1
                                                #define EC_SYNCO_DIS 0x2

// CAP/APWM mode bit #define EC_CAP_MODE 0x0    #define EC_APWM_MODE 0x1
// APWMPOL bit       #define EC_ACTV_HI 0x0     #define EC_ACTV_LO 0x1
// Generic            #define EC_DISABLE 0x0    #define EC_ENABLE 0x1 #define EC_FORCE 0x1

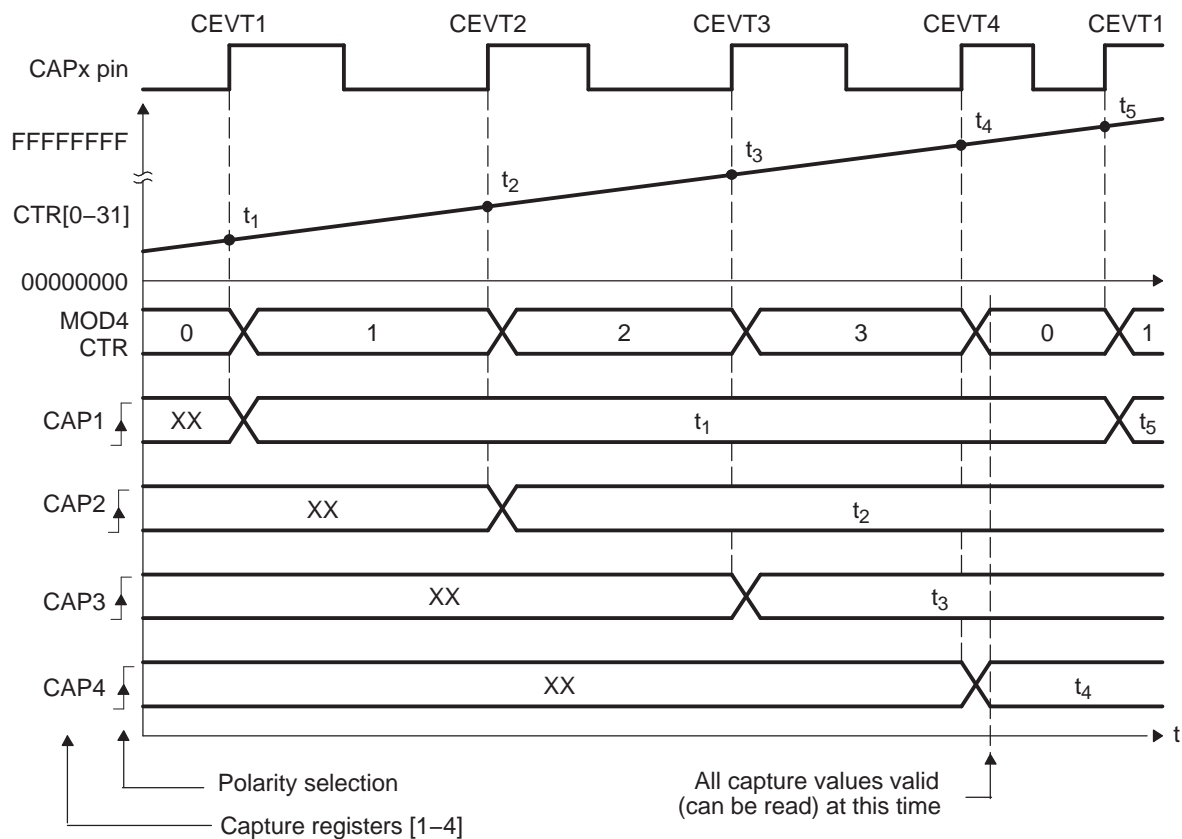
```

15.6.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger

Figure 15-11 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 15-11), 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 15-11. Capture Sequence for Absolute Time-stamp and Rising Edge Detect



15.6.1.1 Code snippet for CAP mode Absolute Time, Rising Edge Trigger

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

// Initialization Time
//=====

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;

ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECap1Regs.ECCTL2.bit.SYNCl_EN = EC_DISABLE;
ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

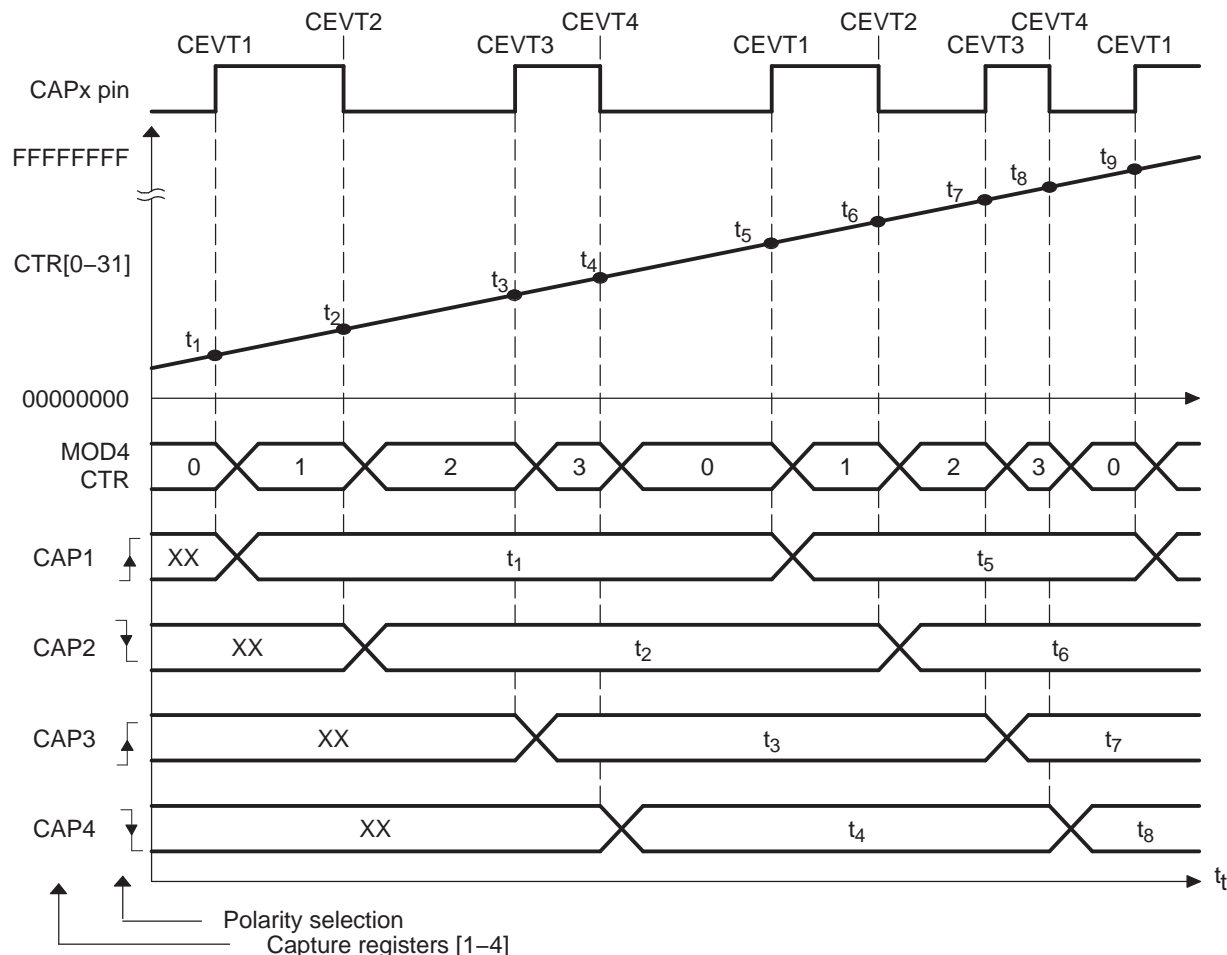
// Run Time (CEVT4 triggered ISR call)
//=====

TSt1 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt2-TSt1;
// Calculate 1st period Period2 = TSt3-TSt2;
// Calculate 2nd period Period3 = TSt4-TSt3;
// Calculate 3rd period
```


15.6.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 15-12 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, i.e: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, ...etc. Duty Cycle1 (on-time %) = $(t_2 - t_1) / \text{Period1} \times 100\%$, etc. Duty Cycle1 (off-time %) = $(t_3 - t_2) / \text{Period1} \times 100\%$, etc.

Figure 15-12. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect



15.6.2.1 Code Snippet for CAP mode Absolute Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Absolute Time, Rising and Falling

// edge triggers // Initialization Time

//=====

// ECAP module 1 config ECAP1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
ECAP1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
ECAP1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
ECAP1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECAP1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
ECAP1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECAP1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECAP1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECAP1Regs.ECCTL2.bit.SYNCl_EN = EC_DISABLE;
ECAP1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT4 triggered ISR call)

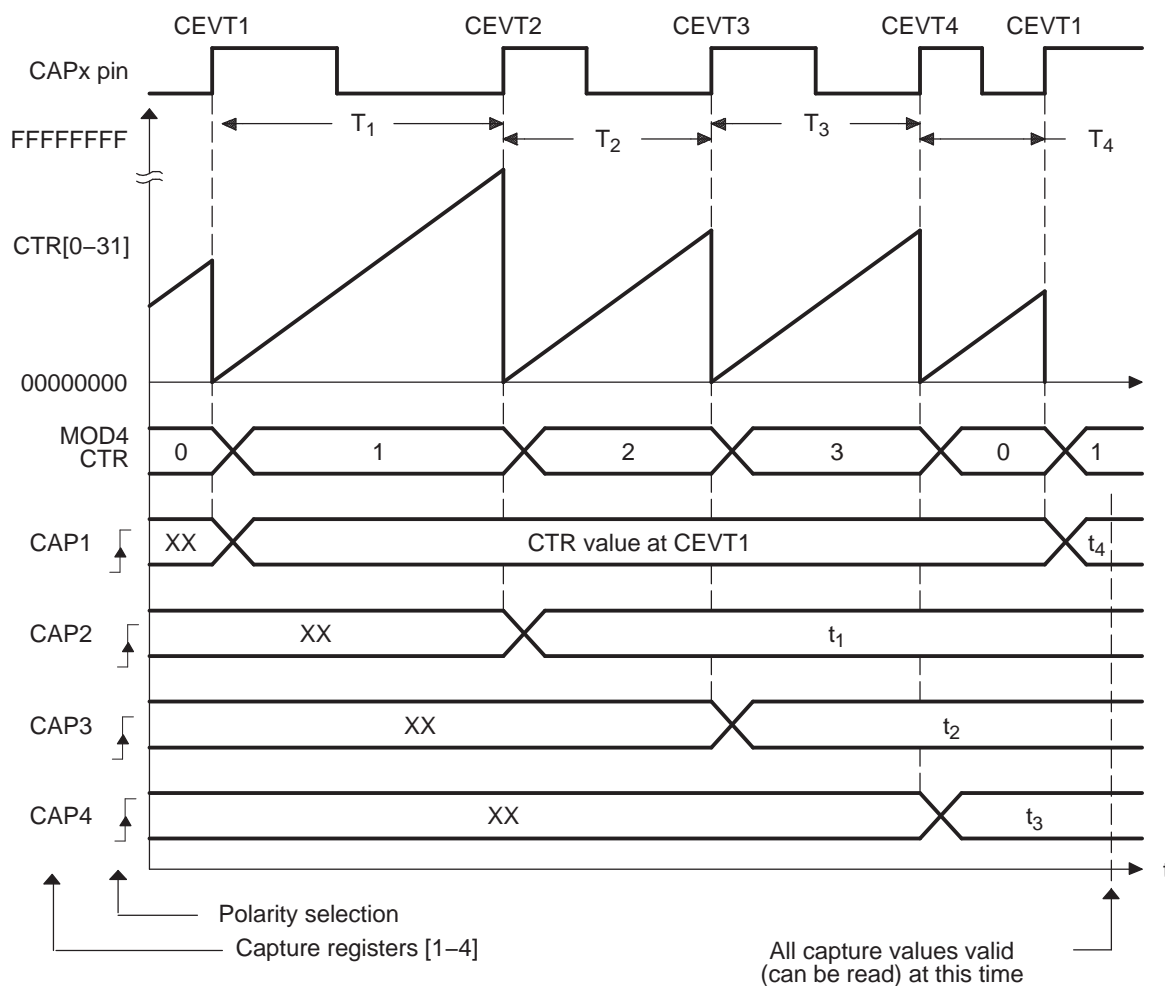
//=====

TSt1 = ECAP1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECAP1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECAP1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECAP1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt3-TSt1;
// Calculate 1st period DutyOnTime1 = TSt2-TSt1;
// Calculate On time DutyOffTime1 = TSt3-TSt2;
// Calculate Off time
```

15.6.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger

This example Figure 15-13 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 CINTOVF (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 ,...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 15-13. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect



15.6.3.1 Code snippet for CAP mode Delta Time, Rising Edge Trigger

```
// Code snippet for CAP mode Delta Time, Rising edge trigger

// Initialization Time

//=====

// ECAP module 1 config ECAP1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECAP1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
ECAP1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECAP1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECAP1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECAP1Regs.ECCTL2.bit.SYNCl_EN = EC_DISABLE;
ECAP1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

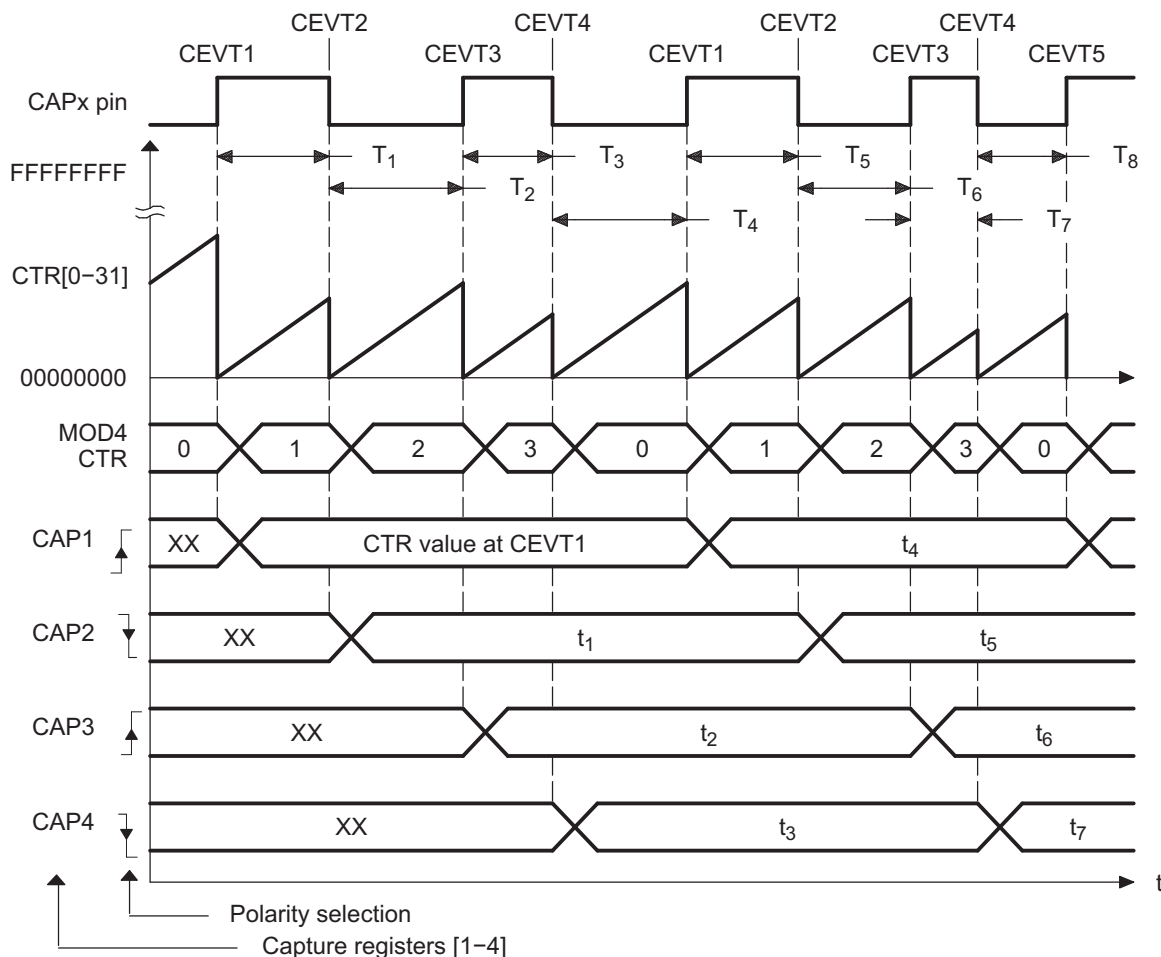
//=====

// Note: here Time-stamp directly represents the Period value. Period4 = ECAP1Regs.CAP1;
// Fetch Time-Stamp captured at T1 Period1 = ECAP1Regs.CAP2;
// Fetch Time-Stamp captured at T2 Period2 = ECAP1Regs.CAP3;
// Fetch Time-Stamp captured at T3 Period3 = ECAP1Regs.CAP4;
// Fetch Time-Stamp captured at T4
```

15.6.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 15-14 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, i.e: $\text{Period1} = T_1 + T_2$, $\text{Period2} = T_3 + T_4$, ...etc $\text{Duty Cycle1 (on-time \%)} = T_1 / \text{Period1} \times 100\%$, etc $\text{Duty Cycle1 (off-time \%)} = T_2 / \text{Period1} \times 100\%$, etc

Figure 15-14. 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 will then automatically copy the init values into the shadow values. For subsequent compare updates, during run-time, only the shadow registers must be used.

15.6.4.1 Code snippet for CAP mode Delta Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Delta Time, Rising and Falling
// edge triggers
// Initialization Time

//=====

// ECAP module 1 config ECAP1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
ECAP1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECAP1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
ECAP1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
ECAP1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECAP1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
ECAP1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECAP1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECAP1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECAP1Regs.ECCTL2.bit.SYNCl_EN = EC_DISABLE;
ECAP1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

//=====
//
Note: here Time-stamp directly represents the Duty cycle values. DutyOnTime1 = ECAP1Regs.CAP2;

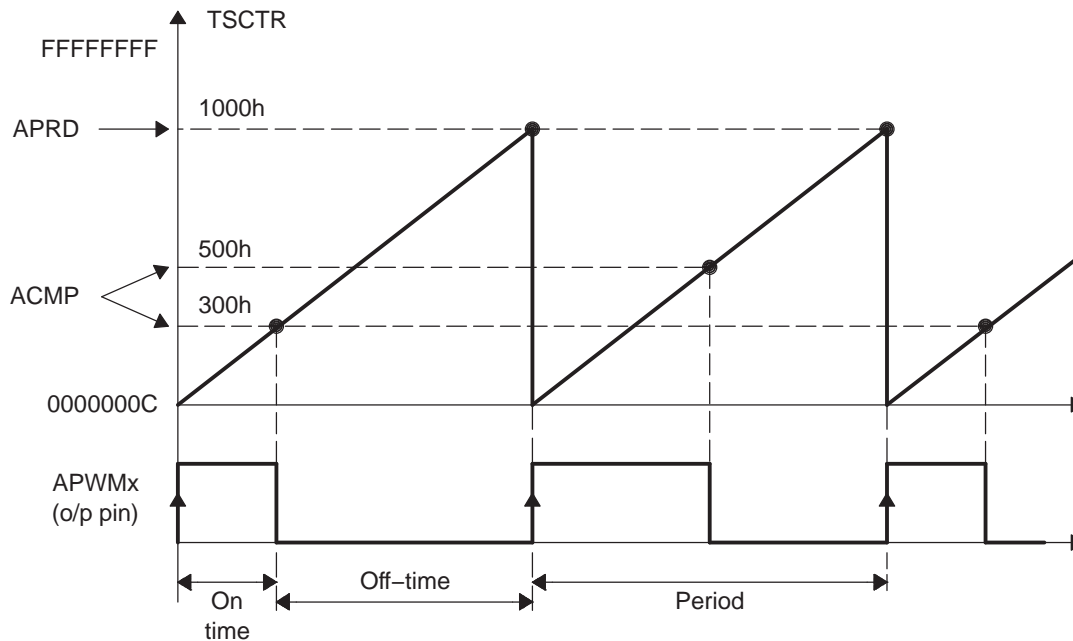
// Fetch Time-Stamp captured at T2 DutyOffTime1 = ECAP1Regs.CAP3;
// Fetch Time-Stamp captured at T3 DutyOnTime2 = ECAP1Regs.CAP4;
// Fetch Time-Stamp captured at T4 DutyOffTime2 = ECAP1Regs.CAP1;
// Fetch Time-
Stamp captured at T1 Period1 = DutyOnTime1 + DutyOffTime1; Period2 = DutyOnTime2 + DutyOffTime2;
```

15.7 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 output pin APWMx. 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. Note here values are in hexadecimal ("h") notation.

15.7.1 Example 1 - Simple PWM Generation (Independent Channel/s)

Figure 15-15. PWM Waveform Details of APWM Mode Operation



Example 15-1. Code Snippet for APWM Mode

```
// Code snippet for APWM mode Example 1

// Initialization Time

//=====

// ECAP module 1 config ECAP1Regs.CAP1 = 0x1000;

// Set period value ECAP1Regs.CTRPHS = 0x0;

// make phase zero ECAP1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
// ECAP1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;

// Active high ECAP1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;

// Synch not used ECAP1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;

// Synch not used ECAP1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (Instant 1, for example, ISR call)

//=====
ECAP1Regs.CAP2 = 0x300;
```

Example 15-1. Code Snippet for APWM Mode (continued)

```
// Set Duty cycle, that is, compare value

// Run Time (Instant 2, for example, another ISR call)

//=====\
    ECAP1Regs.CAP2 = 0x500;

// Set Duty cycle, that is, compare value
```


15.8 Registers

15.8.1 eCAP Base Addresses

Table 15-1. eCAP Base Address Table

Device Register	Register Name	Start Address	End Address
ECap1Regs	ECAP_REGS	0x0000_5000	0x0000_501F
ECap2Regs	ECAP_REGS	0x0000_5020	0x0000_503F
ECap3Regs	ECAP_REGS	0x0000_5040	0x0000_505F
ECap4Regs	ECAP_REGS	0x0000_5060	0x0000_507F
ECap5Regs	ECAP_REGS	0x0000_5080	0x0000_509F
ECap6Regs	ECAP_REGS	0x0000_50A0	0x0000_50BF

15.8.2 ECAP_REGS Registers

Table 15-2 lists the memory-mapped registers for the ECAP_REGS. All register offset addresses not listed in Table 15-2 should be considered as reserved locations and the register contents should not be modified.

Table 15-2. ECAP_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	TSCTR	Time-Stamp Counter		Go
2h	CTRPHS	Counter Phase Offset Value Register		Go
4h	CAP1	Capture 1 Register		Go
6h	CAP2	Capture 2 Register		Go
8h	CAP3	Capture 3 Register		Go
Ah	CAP4	Capture 4 Register		Go
14h	ECCTL1	Capture Control Register 1		Go
15h	ECCTL2	Capture Control Register 2		Go
16h	ECEINT	Capture Interrupt Enable Register		Go
17h	ECFLG	Capture Interrupt Flag Register		Go
18h	ECCLR	Capture Interrupt Flag Register		Go
19h	ECFRC	Capture Interrupt Force Register		Go

15.8.2.1 TSCTR Register (Offset = 0h) [reset = 0h]

TSCTR is shown in [Figure 15-16](#) and described in [Table 15-3](#).

Time-Stamp Counter

Figure 15-16. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-3. 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

15.8.2.2 CTRPHS Register (Offset = 2h) [reset = 0h]

CTRPHS is shown in [Figure 15-17](#) and described in [Table 15-4](#).

Counter Phase Offset Value Register

Figure 15-17. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-4. CTRPHS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CTRPHS	R/W	0h	Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCTR and 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.

15.8.2.3 CAP1 Register (Offset = 4h) [reset = 0h]

CAP1 is shown in [Figure 15-18](#) and described in [Table 15-5](#).

Capture 1 Register

Figure 15-18. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-5. CAP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP1	R/W	0h	This register can be loaded (written) by :) Time-Stamp (counter value TSCTR) during a capture event) Software - may be useful for test purposes / initialization) APRD shadow register (CAP3) when used in APWM mode

15.8.2.4 CAP2 Register (Offset = 6h) [reset = 0h]

CAP2 is shown in [Figure 15-19](#) and described in [Table 15-6](#).

Capture 2 Register

Figure 15-19. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-6. 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"> - Time-Stamp (counter value) during a capture event - Software - may be useful for test purposes - APRD shadow register (CAP4) when used in APWM mode

15.8.2.5 CAP3 Register (Offset = 8h) [reset = 0h]

CAP3 is shown in [Figure 15-20](#) and described in [Table 15-7](#).

Capture 3 Register

Figure 15-20. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-7. 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 update the PWM period value through this register. In this mode, CAP3 (APRD) shadows CAP1.

15.8.2.6 CAP4 Register (Offset = Ah) [reset = 0h]

CAP4 is shown in [Figure 15-21](#) and described in [Table 15-8](#).

Capture 4 Register

Figure 15-21. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-8. CAP4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP4	R/W	0h	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. You update the PWM compare value via this register. In this mode, CAP4 (ACMP) shadows CAP2.

15.8.2.7 ECCTL1 Register (Offset = 14h) [reset = 0h]

ECCTL1 is shown in [Figure 15-22](#) and described in [Table 15-9](#).

Capture Control Register 1

Figure 15-22. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-9. ECCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Control 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 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. 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 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 0h (R/W) = Capture Event 4 triggered on a rising edge (RE) 1h (R/W) = Capture Event 4 triggered on a falling edge (FE)
5	CTRRST3	R/W	0h	Counter Reset on Capture Event 3 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 0h (R/W) = Capture Event 3 triggered on a rising edge (RE) 1h (R/W) = Capture Event 3 triggered on a falling edge (FE)

Table 15-9. ECCTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	CTRRST2	R/W	0h	Counter Reset on Capture Event 2 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 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 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 0h (R/W) = Capture Event 1 triggered on a rising edge (RE) 1h (R/W) = Capture Event 1 triggered on a falling edge (FE)

15.8.2.8 ECCTL2 Register (Offset = 15h) [reset = 2h]

ECCTL2 is shown in [Figure 15-23](#) and described in [Table 15-10](#).

Capture Control Register 2

Figure 15-23. ECCTL2 Register

15	14	13	12	11	10	9	8
RESERVED					APWMPOL	CAP_APWM	SWSYNC
R-0h					R/W-0h	R/W-0h	R/W-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/W-0h	R/W-1h		R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-10. ECCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	APWMPOL	R/W	0h	APWM output polarity select. 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 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
8	SWSYNC	R/W	0h	Software-forced Counter (TSCTR) Synchronizing. 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 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

Table 15-10. ECCTL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	SYNCL_EN	R/W	0h	Counter (TSCTR) Sync-In select mode 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 0h (R/W) = TSCTR stopped 1h (R/W) = TSCTR free-running
3	REARM	R/W	0h	Re-Arming Control. Note: The re-arm function is valid in one shot or continuous mode. 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
2-1	STOP_WRAP	R/W	1h	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. 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. Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur: - Mod4 counter is stopped (frozen) - Capture register loads are inhibited In one-shot mode, further interrupt events are blocked until re-armed. 0h (R/W) = Stop after Capture Event 1 in one-shot mode Wrap after Capture Event 1 in continuous mode. 1h (R/W) = Stop after Capture Event 2 in one-shot mode Wrap after Capture Event 2 in continuous mode. 2h (R/W) = Stop after Capture Event 3 in one-shot mode Wrap after Capture Event 3 in continuous mode. 3h (R/W) = Stop after Capture Event 4 in one-shot mode Wrap after Capture Event 4 in continuous mode.
0	CONT_ONESHT	R/W	0h	Continuous or one-shot mode control (applicable only in capture mode) 0h (R/W) = Operate in continuous mode 1h (R/W) = Operate in one-Shot mode

15.8.2.9 ECEINT Register (Offset = 16h) [reset = 0h]

ECEINT is shown in [Figure 15-24](#) and described in [Table 15-11](#).

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 15-24. ECEINT Register

15	14	13	12	11	10	9	8
RESERVED							
R/W-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/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-11. ECEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	CTR_EQ_CMP	R/W	0h	Counter Equal Compare Interrupt Enable 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 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 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 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 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 0h (R/W) = Disable Capture Event 2 as an Interrupt source 1h (R/W) = Enable Capture Event 2 as an Interrupt source
1	CEVT1	R/W	0h	Capture Event 1 Interrupt Enable 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/W	0h	Reserved

15.8.2.10 ECFLG Register (Offset = 17h) [reset = 0h]

ECFLG is shown in [Figure 15-25](#) and described in [Table 15-12](#).

Capture Interrupt Flag Register

Figure 15-25. ECFLG Register

15	14	13	12	11	10	9	8
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-12. ECFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	CTR_CMP	R	0h	Compare Equal Compare Status Flag. This flag is active only in APWM mode. 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. 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. 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) has made the transition from FFFFFFFF " 00000000
4	CEVT4	R	0h	Capture Event 4 Status Flag This flag is only active in CAP mode. 0h (R/W) = Indicates no event occurred 1h (R/W) = ndicates the fourth event occurred at ECAPx pin
3	CEVT3	R	0h	Capture Event 3 Status Flag. This flag is active only in CAP mode. 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. 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. 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the first event occurred at ECAPx pin.
0	INT	R	0h	Global Interrupt Status Flag 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates that an interrupt was generated.

15.8.2.11 ECCLR Register (Offset = 18h) [reset = 0h]

ECCLR is shown in [Figure 15-26](#) and described in [Table 15-13](#).

Capture Interrupt Flag Register

Figure 15-26. ECCLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-13. ECCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	CTR_CMP	R/W	0h	Counter Equal Compare Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTR=CMF flag condition
6	CTR_PRD	R/W	0h	Counter Equal Period Status Flag 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 condition
5	CTROVF	R/W	0h	Counter Overflow Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTROVF flag condition
4	CEVT4	R/W	0h	Capture Event 4 Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT4 flag condition.
3	CEVT3	R/W	0h	Capture Event 3 Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT3 flag condition.
2	CEVT2	R/W	0h	Capture Event 2 Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT2 flag condition.
1	CEVT1	R/W	0h	Capture Event 1 Status Flag 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT1 flag condition.
0	INT	R/W	0h	Global Interrupt Clear Flag 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

15.8.2.12 ECFRC Register (Offset = 19h) [reset = 0h]

ECFRC is shown in [Figure 15-27](#) and described in [Table 15-14](#).

Capture Interrupt Force Register

Figure 15-27. ECFRC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 15-14. ECFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	CTR_CMP	R/W	0h	Force Counter Equal Compare Interrupt. This event is only active in APWM mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR=CMP flag bit.
6	CTR_PRD	R/W	0h	Force Counter Equal Period Interrupt. This event is only active in APWM mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR=PRD flag bit.
5	CTROVF	R/W	0h	Force Counter Overflow. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 to this bit sets the CTROVF flag bit.
4	CEVT4	R/W	0h	Force Capture Event 4. This event is only active in CAP mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT4 flag bit
3	CEVT3	R/W	0h	Force Capture Event 3. This event is only active in CAP mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT3 flag bit
2	CEVT2	R/W	0h	Force Capture Event 2. This event is only active in CAP mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT2 flag bit.
1	CEVT1	R/W	0h	Force Capture Event 1. This event is only active in CAP mode. 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Sets the CEVT1 flag bit.
0	RESERVED	R	0h	Reserved

Enhanced QEP (eQEP)

The enhanced QEP (eQEP) module described here is a Type 0 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.

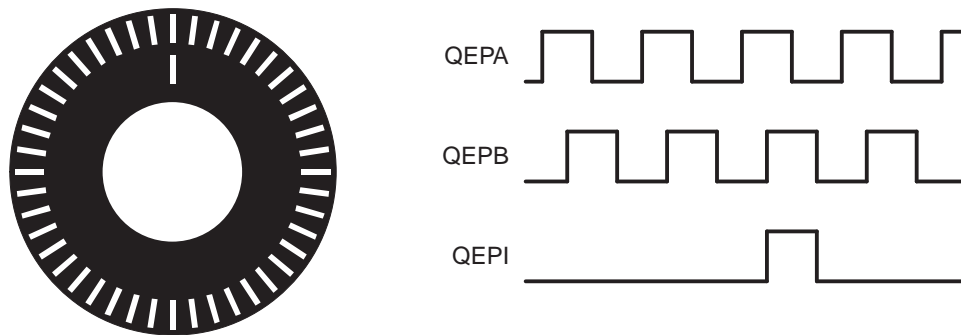
The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.

Topic	Page
16.1 Introduction	1824
16.2 Configuring Device Pins	1826
16.3 Description	1826
16.4 Quadrature Decoder Unit (QDU).....	1829
16.5 Position Counter and Control Unit (PCCU).....	1832
16.6 eQEP Edge Capture Unit.....	1839
16.7 eQEP Watchdog.....	1842
16.8 Unit Timer Base	1842
16.9 eQEP Interrupt Structure	1843
16.10 Registers	1843

16.1 Introduction

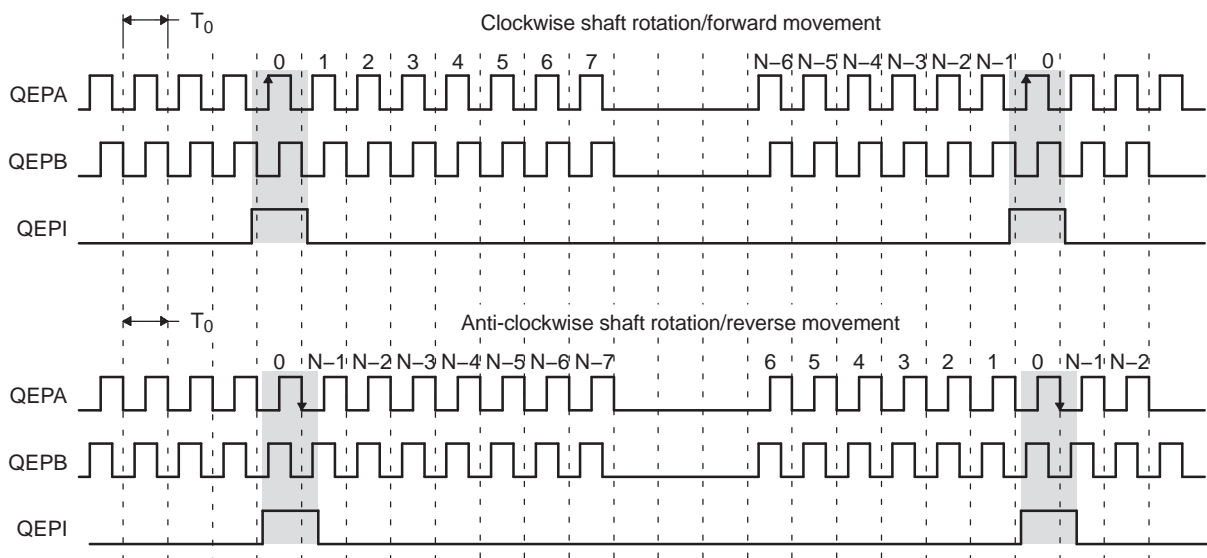
A single track of slots patterns the periphery of an incremental encoder disk, as shown in [Figure 16-1](#). These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, 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 16-1. 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 realized 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 16-2](#).

Figure 16-2. QEP Encoder Output Signal for Forward/Reverse Movement

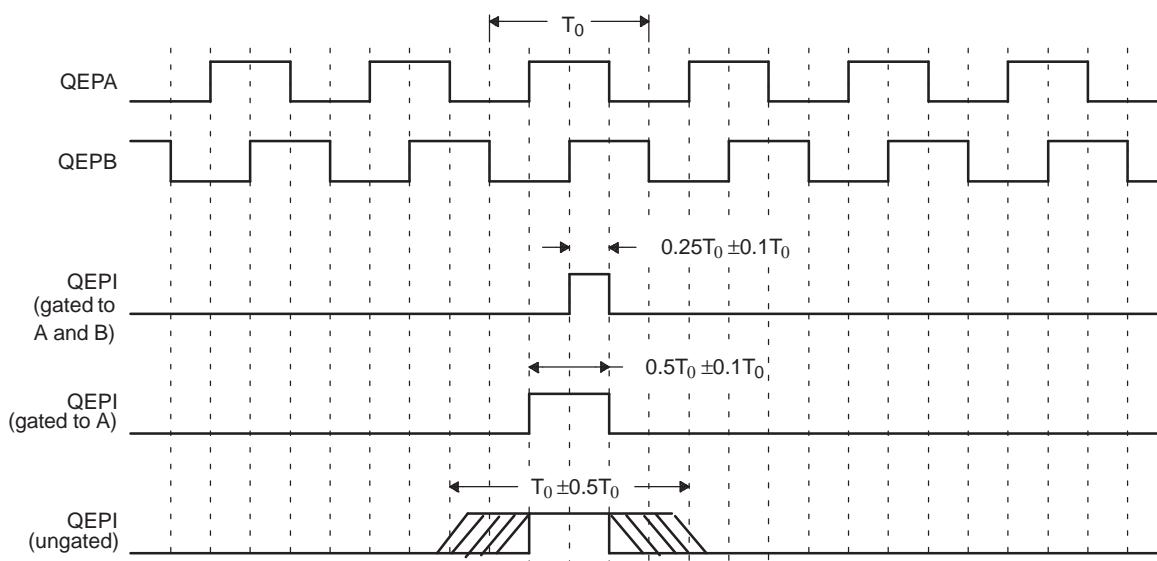


Legend: N = lines per revolution

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 16-3. 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 16-3. Index Pulse Example



Some typical applications of shaft encoders include robotics and even 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:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (1)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (2)$$

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 1 is the conventional approach to velocity estimation and it requires a time base to provide 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 1](#) 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, e.g. 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 2](#) 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 2](#) 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 1](#). A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT 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 2](#) at low speed and have the DSP software switch over to [Equation 1](#) when the motor speed rises above some specified threshold.

16.2 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.

16.3 Description

This section provides the eQEP inputs, memory map, and functional description.

16.3.1 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 *System Control and Interrupts* 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 whose 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*

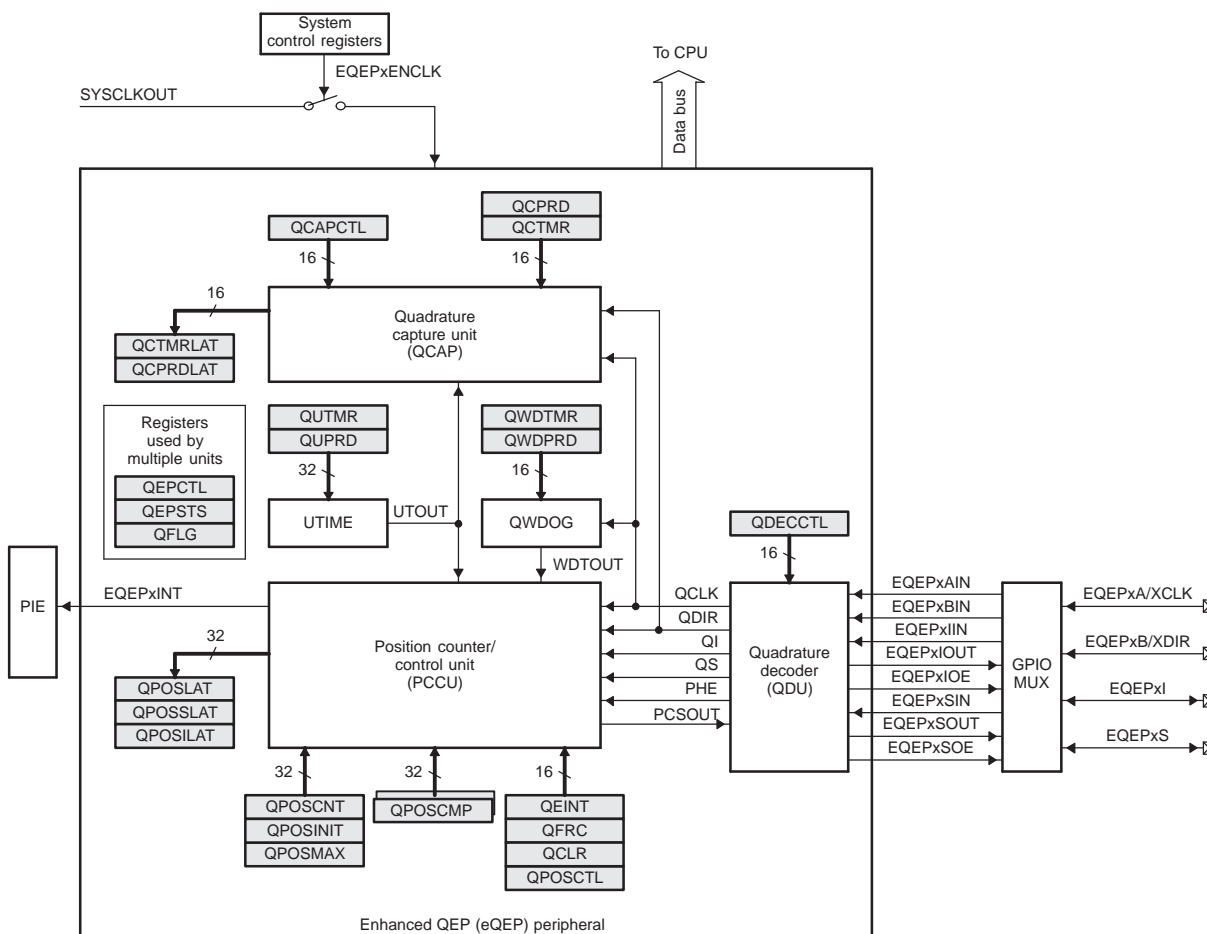
This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

16.3.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in [Figure 16-4](#)):

- 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)

Figure 16-4. Functional Block Diagram of the eQEP Peripheral



16.3.3 eQEP Memory Map

Table 16-1 lists the registers with their memory locations, sizes, and reset values.

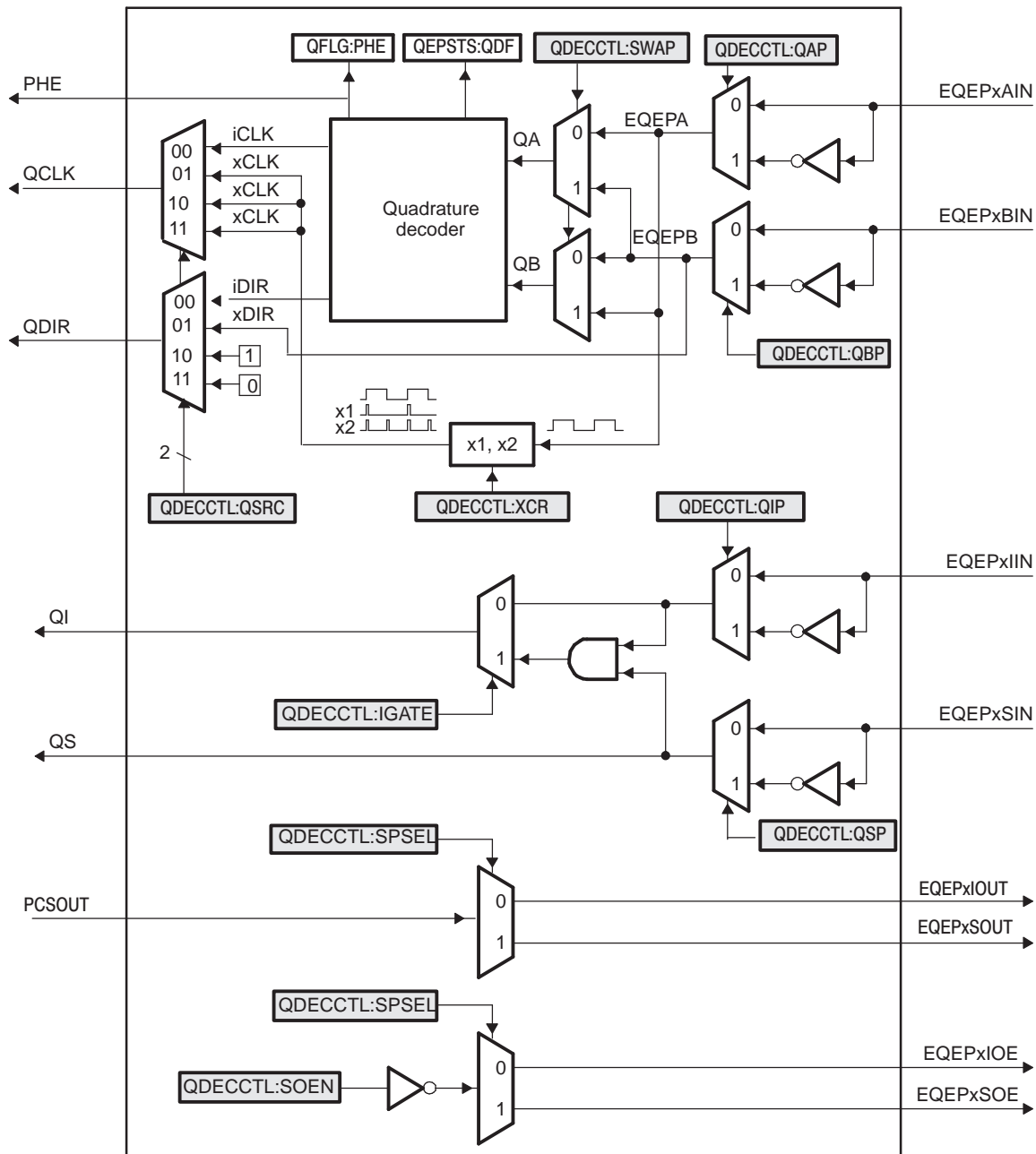
Table 16-1. EQEP Memory Map

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	0x0000	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 0x3F	31/0		

16.4 Quadrature Decoder Unit (QDU)

Figure 16-5 shows a functional block diagram of the QDU.

Figure 16-5. Functional Block Diagram of Decoder Unit



16.4.1 Position Counter Input Modes

Clock and direction input to 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

16.4.1.1 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 QEPSTS[QDF] bit. Table 16-2 and Figure 16-6 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 16-7 shows the direction decoding and clock generation from the eQEP input signals.

Table 16-2. 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 16-6. Quadrature Decoder State Machine

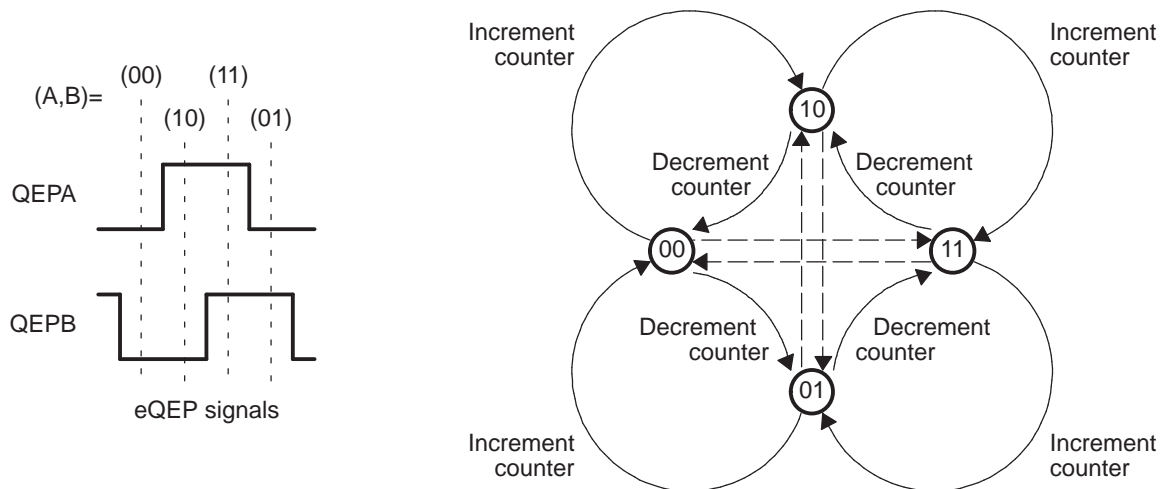
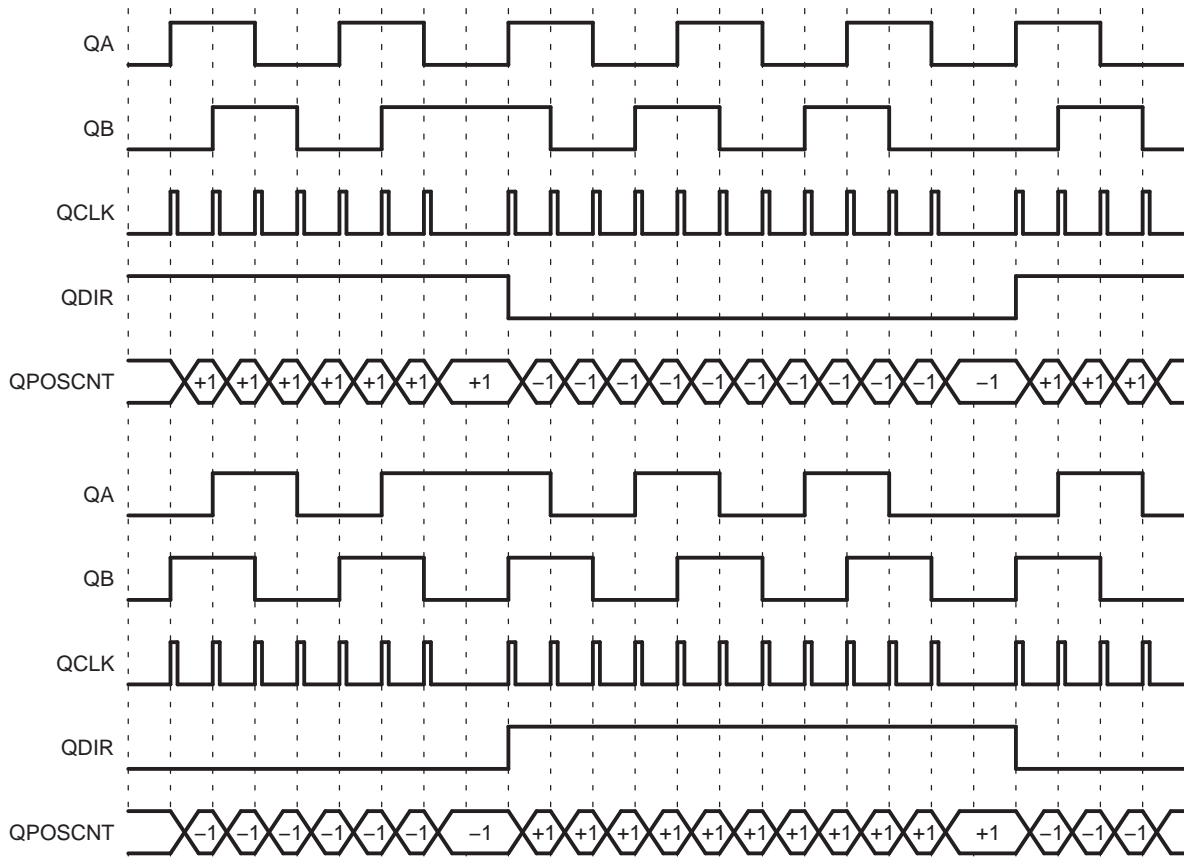


Figure 16-7. Quadrature-clock and Direction Decoding



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 when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 16-6 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 16-7.

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.

16.4.1.2 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 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.

16.4.1.3 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 2x factor. 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.

16.4.1.4 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 of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2x factor. 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.

16.4.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting of QDECCTL[QIP] bit will invert the index input.

16.4.3 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.

16.5 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.

16.5.1 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 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 position counter provides rotor angle with respect to index pulse position.

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, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in QFLG register.

16.5.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM]=00)

If the index event occurs during the forward movement, then 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.

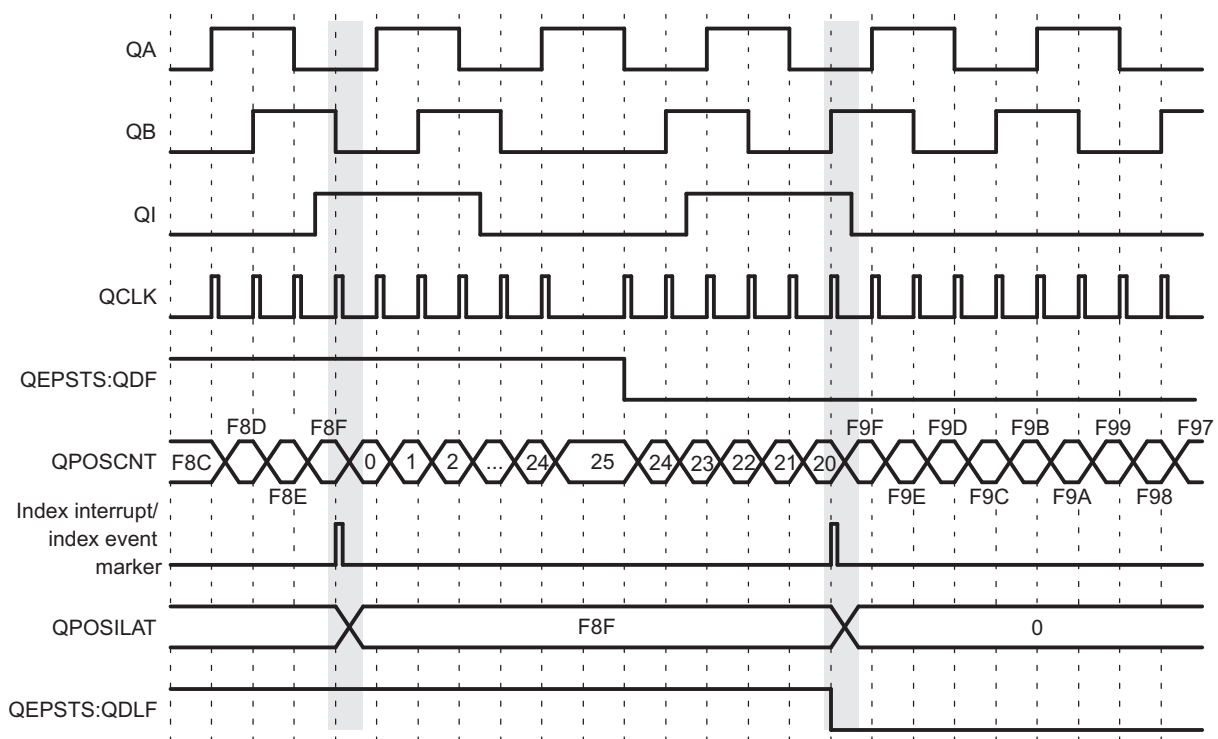
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first 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 16-8.

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 (QFLG[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 (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

Figure 16-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F)

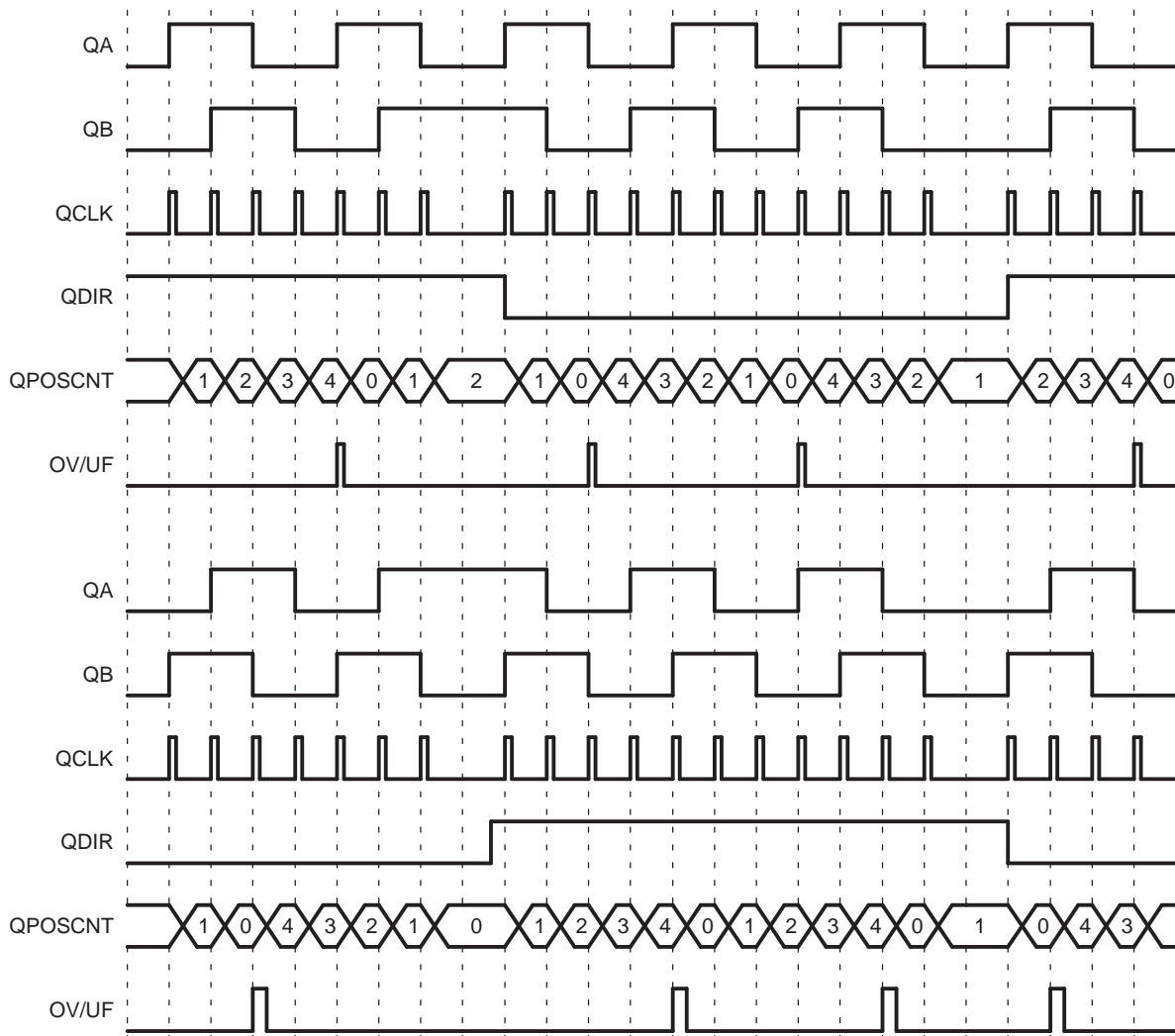


16.5.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM]=01)

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 16-9 shows the position counter reset operation in this mode.

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 the same relative quadrature transition is used for the software index marker (QEPCTL[IEL]=11).

Figure 16-9. Position Counter Underflow/Overflow (QPOSMAX = 4)



16.5.1.3 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 [Section 16.5.1.2](#).

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 software index marker (QEPCTL[IEL]=11).

16.5.1.4 Position Counter Reset on Unit Time out Event (QEPCTL[PCRM] = 11)

In this mode, the QPOSCNT value is latched to the QPOSLAT register and then the QPOSCNT is reset (to 0 or QPOSMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event). This is useful for frequency measurement.

16.5.2 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.

16.5.2.1 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.

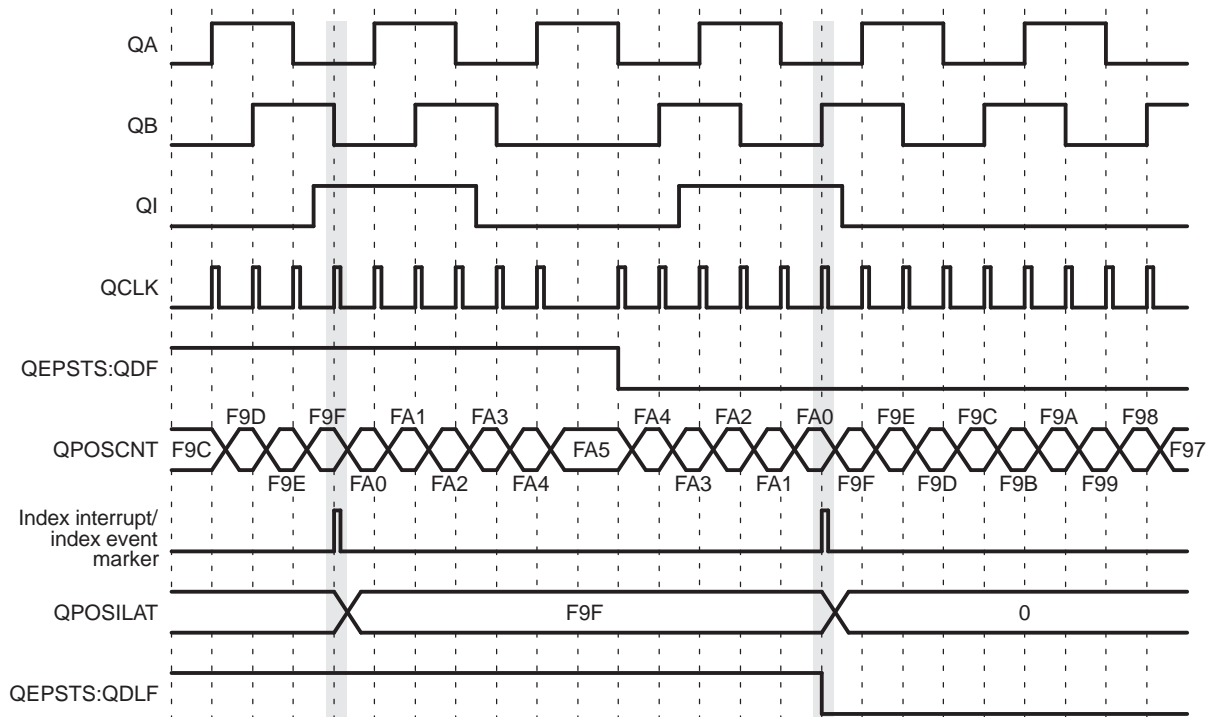
The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register.

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 16-10 shows the position counter latch using an index event marker.

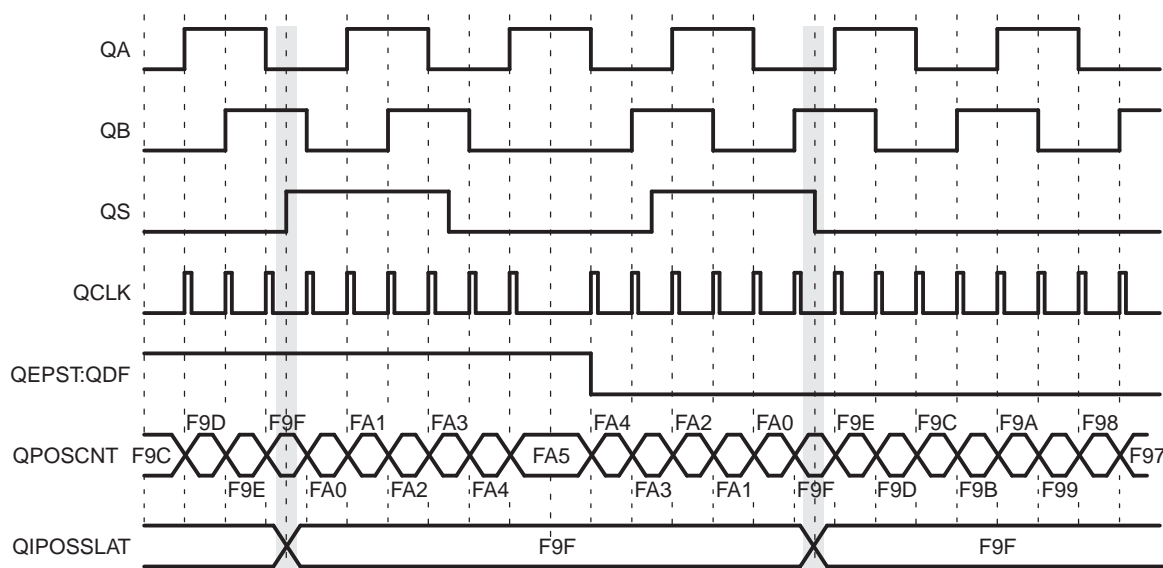
Figure 16-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)


16.5.2.2 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 16-11](#).

The strobe event latch interrupt flag (QFLG[SEL]) is set when the position counter is latched to the QPOSSLAT register.

Figure 16-11. Strobe Event Latch (QEPCTL[SEL] = 1)


16.5.3 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.

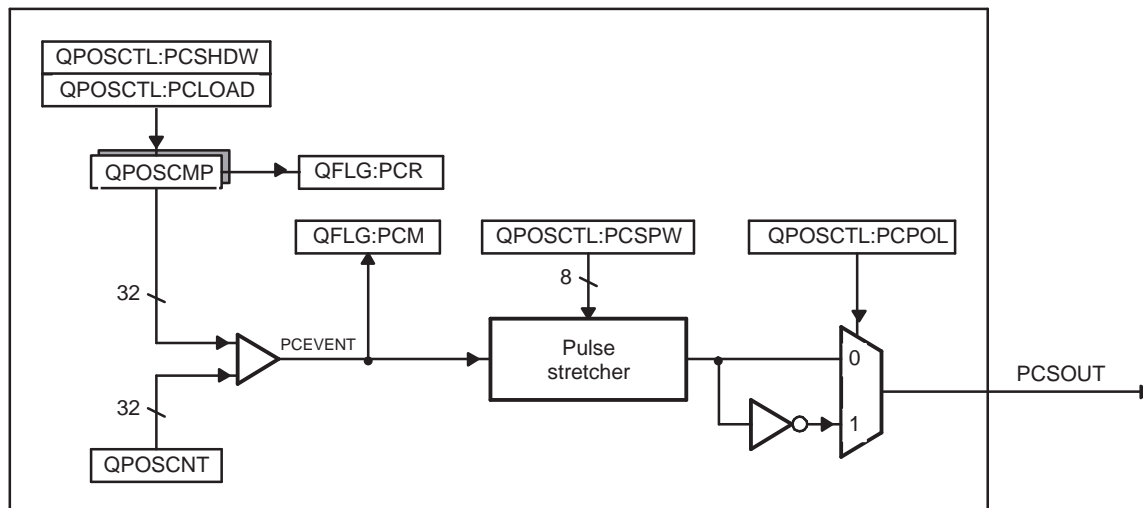
If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

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.

16.5.4 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 16-12 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 16-12. 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.

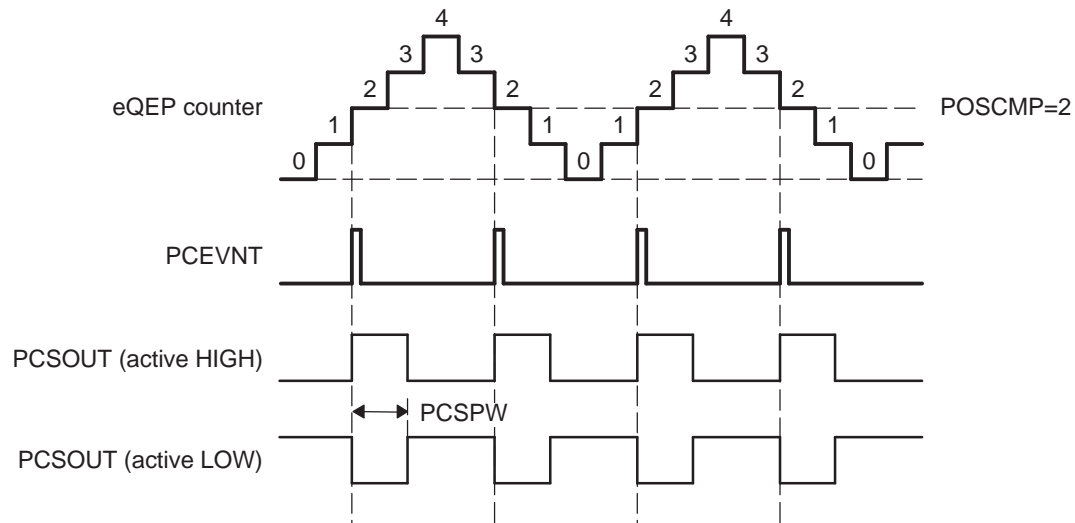
- Load on compare match
- Load on position-counter zero event

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 16-13](#)).

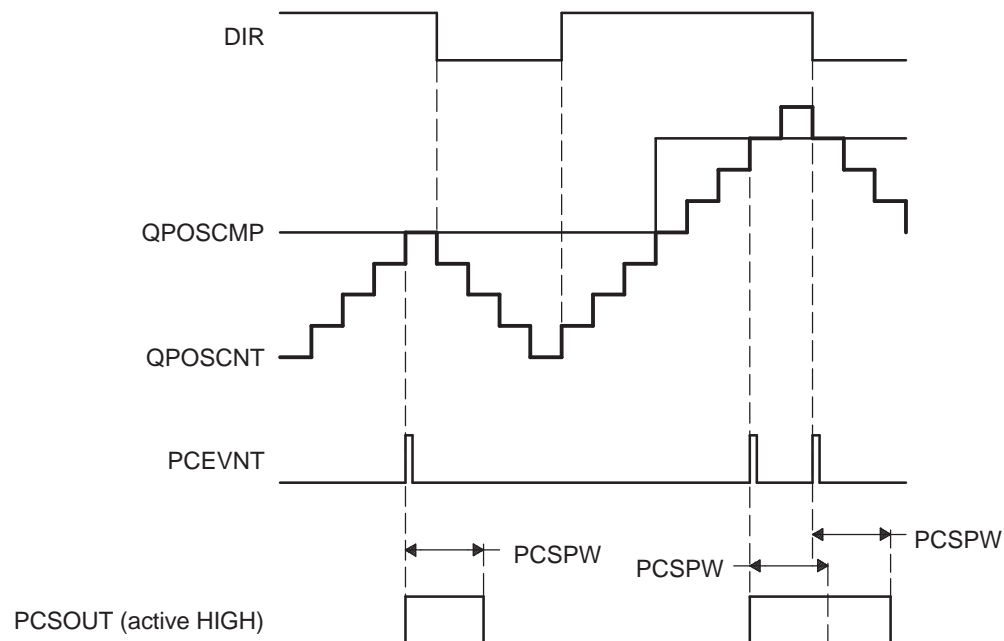
See the register section for the layout of the eQEP Position-Compare Control Register (QPOSCTL) and description of the QPOSCTL bit fields.

Figure 16-13. 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 16-14](#).

Figure 16-14. eQEP Position-compare Sync Output Pulse Stretcher



16.6 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 16-15](#). This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k - 1)} = \frac{X}{\Delta T} \quad (3)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see [Figure 16-16](#))
- ΔT - 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 (ΔT) 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]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

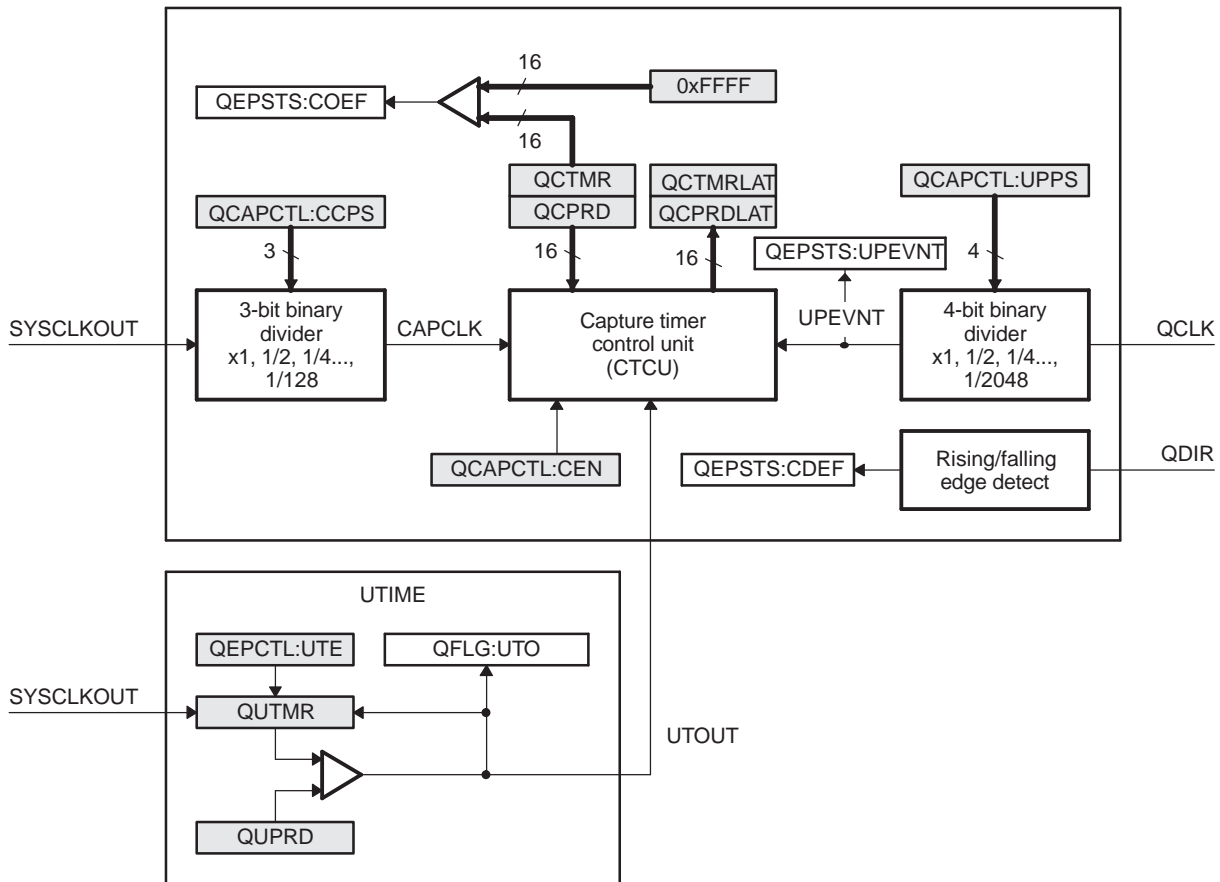
- CPU read of QPOSCNT register
- Unit time-out event

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.

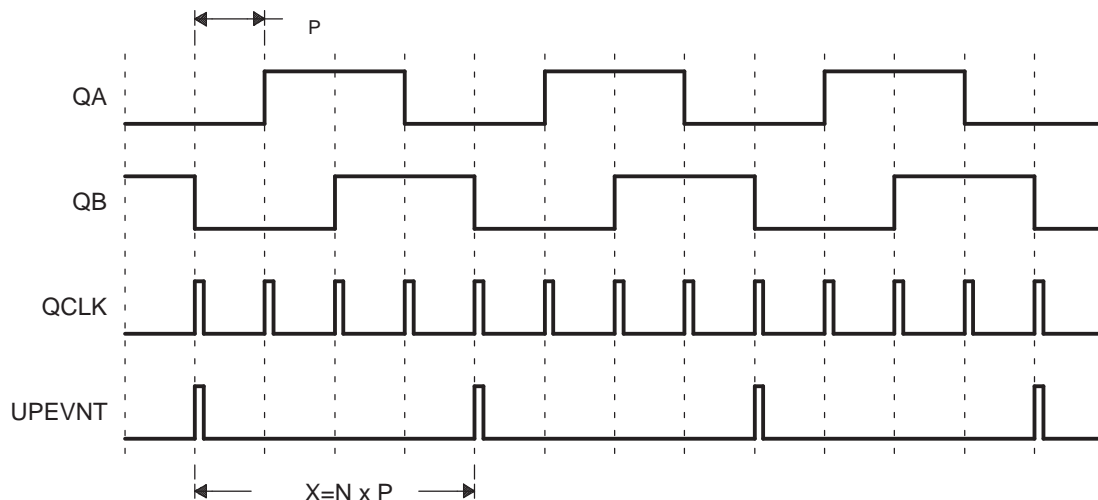
[Figure 16-17](#) shows the capture unit operation along with the position counter.

Figure 16-15. eQEP Edge Capture Unit

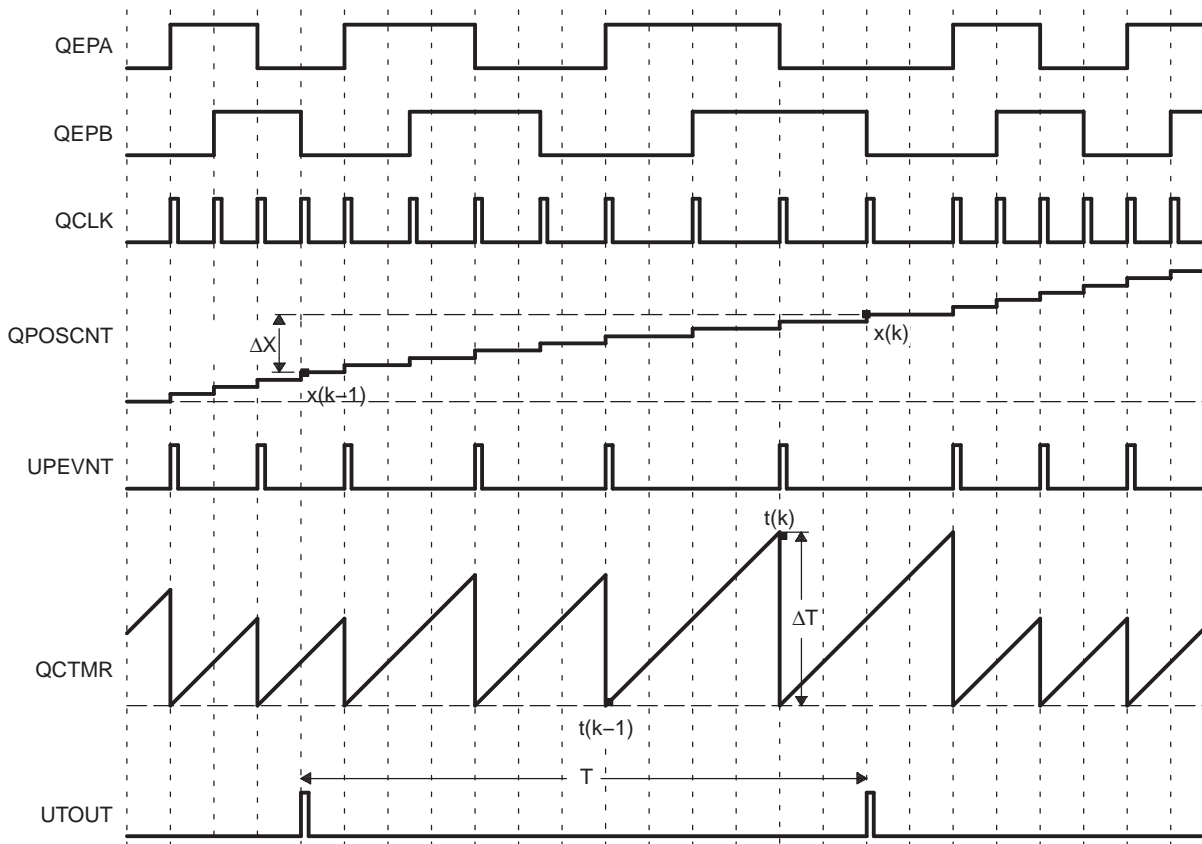


NOTE: The **QCAPCTL[UPPS]** prescaler should not be modified dynamically (such as switching the unit event prescaler from **QCLK/4** to **QCLK/8**). Doing so may result in undefined behavior. The **QCAPCTL[CPPS]** prescaler can be modified dynamically (such as switching **CAPCLK** prescaling mode from **SYSCLK/4** to **SYSCLK/8**) only after the capture unit is disabled.

Figure 16-16. Unit Position Event for Low Speed Measurement (**QCAPCTL[UPPS] = 0010**)



A **N** - Number of quadrature periods selected using **QCAPCTL[UPPS]** bits

Figure 16-17. eQEP Edge Capture Unit - Timing Details


Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (4)$$

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

X : Fixed unit position

ΔT : Incremental time elapsed for unit position movement

$t(k)$: Time instant " k "

$t(k-1)$: Time instant " $k-1$ "

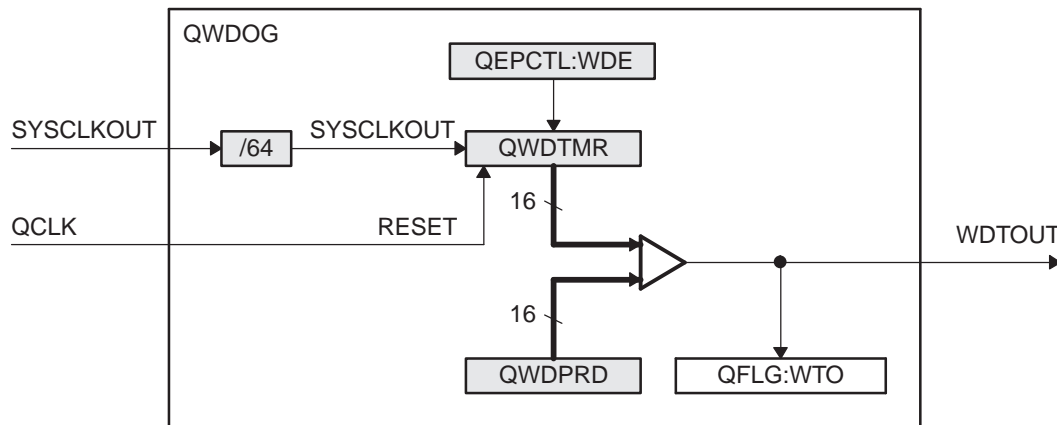
Unit time (T) and unit period(X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (QUPRD)
ΔX	Incremental Position = QPOSAT(k) - QPOSAT(K-1)
X	Fixed unit position defined by sensor resolution and ZCAPCTL[UPPS] bits
ΔT	Capture Period Latch (QCPRDLAT)

16.7 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 16-18. eQEP Watchdog Timer

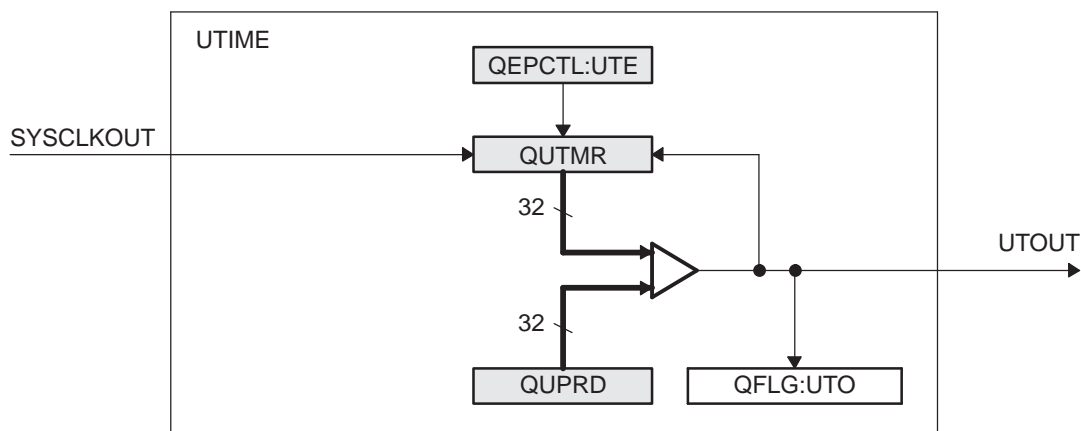


16.8 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD).

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 16.6](#).

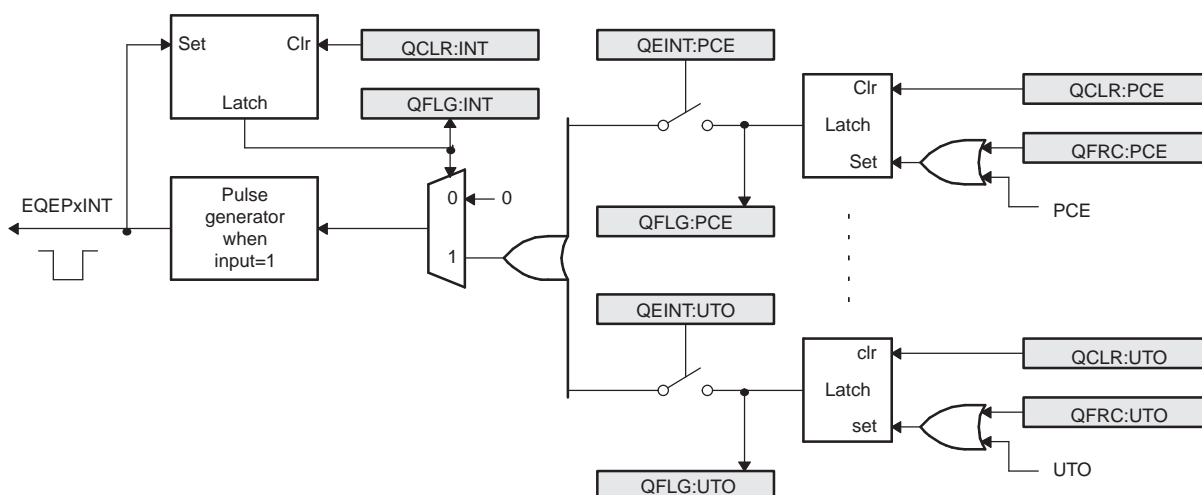
Figure 16-19. eQEP Unit Time Base



16.9 eQEP Interrupt Structure

Figure 16-20 shows how the interrupt mechanism works in the EQEP module.

Figure 16-20. 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 only to the PIE if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. 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. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

16.10 Registers

16.10.1 eQEP Base Addresses

Table 16-3. 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
EQep3Regs	EQEP_REGS	0x0000_5180	0x0000_51BF

16.10.2 EQEP_REGS Registers

Table 16-4 lists the memory-mapped registers for the EQEP_REGS. All register offset addresses not listed in Table 16-4 should be considered as reserved locations and the register contents should not be modified.

Table 16-4. EQEP_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	QPOSCNT	Position Counter		Go
2h	QPOSINIT	Position Counter Init		Go
4h	QPOSMAX	Maximum Position Count		Go
6h	QPOSCMP	Position Compare		Go
8h	QPOSILAT	Index Position Latch		Go
Ah	QPOSSLAT	Strobe Position Latch		Go
Ch	QPOSLAT	Position Latch		Go
Eh	QUTMR	QEP Unit Timer		Go
10h	QUPRD	QEP Unit Period		Go
12h	QWDTMR	QEP Watchdog Timer		Go
13h	QWDPRD	QEP Watchdog Period		Go
14h	QDECCTL	Quadrature Decoder Control		Go
15h	QEPCTL	QEP Control		Go
16h	QCAPCTL	Quadrature Capture Control		Go
17h	QPOSCTL	Position Compare Control		Go
18h	QEINT	QEP Interrupt Control		Go
19h	QFLG	QEP Interrupt Flag		Go
1Ah	QCLR	QEP Interrupt Clear		Go
1Bh	QFRC	QEP Interrupt Force		Go
1Ch	QEPSTS	QEP Status		Go
1Dh	QCTMR	QEP Capture Timer		Go
1Eh	QCPRD	QEP Capture Period		Go
1Fh	QCTMRLAT	QEP Capture Latch		Go
20h	QCPRLAT	QEP Capture Period Latch		Go

16.10.2.1 QPOSCNT Register (Offset = 0h) [reset = 0h]

QPOSCNT is shown in [Figure 16-21](#) and described in [Table 16-5](#).

Position Counter

Figure 16-21. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-5. 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.</p>

16.10.2.2 QPOSINIT Register (Offset = 2h) [reset = 0h]

QPOSINIT is shown in [Figure 16-22](#) and described in [Table 16-6](#).

Position Counter Init

Figure 16-22. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-6. QPOSINIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSINIT	R/W	0h	<p>Position Counter Init</p> <p>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.</p>

16.10.2.3 QPOSMAX Register (Offset = 4h) [reset = 0h]

QPOSMAX is shown in [Figure 16-23](#) and described in [Table 16-7](#).

Maximum Position Count

Figure 16-23. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-7. 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.

16.10.2.4 QPOSCMP Register (Offset = 6h) [reset = 0h]

QPOSCMP is shown in [Figure 16-24](#) and described in [Table 16-8](#).

Position Compare

Figure 16-24. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-8. 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.

16.10.2.5 QPOSILAT Register (Offset = 8h) [reset = 0h]

QPOSILAT is shown in [Figure 16-25](#) and described in [Table 16-9](#).

Index Position Latch

Figure 16-25. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-9. 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.

16.10.2.6 QPOSSLAT Register (Offset = Ah) [reset = 0h]

QPOSSLAT is shown in [Figure 16-26](#) and described in [Table 16-10](#).

Strobe Position Latch

Figure 16-26. QPOSSLAT Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-10. 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 strobe event as defined by the QEPCTL[SEL] bits.

16.10.2.7 QPOSLAT Register (Offset = Ch) [reset = 0h]

QPOSLAT is shown in [Figure 16-27](#) and described in [Table 16-11](#).

Position Latch

Figure 16-27. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-11. 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 unit time out event.

16.10.2.8 QUTMR Register (Offset = Eh) [reset = 0h]

QUTMR is shown in [Figure 16-28](#) and described in [Table 16-12](#).

QEP Unit Timer

Figure 16-28. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-12. QUTMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QUTMR	R/W	0h	QEP Unit Timer This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated.

16.10.2.9 QUPRD Register (Offset = 10h) [reset = 0h]

QUPRD is shown in [Figure 16-29](#) and described in [Table 16-13](#).

QEP Unit Period

Figure 16-29. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-13. QUPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QUPRD	R/W	0h	<p>QEP Unit Period</p> <p>This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt. Writes to this register should always be full 32-bit writes.</p>

16.10.2.10 QWDTMR Register (Offset = 12h) [reset = 0h]

QWDTMR is shown in [Figure 16-30](#) and described in [Table 16-14](#).

QEP Watchdog Timer

Figure 16-30. QWDTMR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QWDTMR															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-14. 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 watch dog to detect motor stalls. When this timer value matches with watch dog period value, watch dog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion.</p>

16.10.2.11 QWDPRD Register (Offset = 13h) [reset = 0h]

QWDPRD is shown in [Figure 16-31](#) and described in [Table 16-15](#).

QEP Watchdog Period

Figure 16-31. QWDPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QWDPRD															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-15. QWDPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QWDPRD	R/W	0h	<p>QEP Watchdog Period</p> <p>This register contains the time-out count for the eQEP peripheral watch dog timer.</p> <p>When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated.</p>

16.10.2.12 QDECCTL Register (Offset = 14h) [reset = 0h]

QDECCTL is shown in [Figure 16-32](#) and described in [Table 16-16](#).

Quadrature Decoder Control

Figure 16-32. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-16. QDECCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	QSRC	R/W	0h	Position-counter source selection 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 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 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 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 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 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 0h (R/W) = No effect 1h (R/W) = Negates QEPA input
7	QBP	R/W	0h	QEPB input polarity 0h (R/W) = No effect 1h (R/W) = Negates QEPB input
6	QIP	R/W	0h	QEPI input polarity 0h (R/W) = No effect 1h (R/W) = Negates QEPI input
5	QSP	R/W	0h	QEPS input polarity 0h (R/W) = No effect 1h (R/W) = Negates QEPS input
4-0	RESERVED	R	0h	Reserved

16.10.2.13 QEPCTL Register (Offset = 15h) [reset = 0h]

QEPCTL is shown in [Figure 16-33](#) and described in [Table 16-17](#).

QEP Control

Figure 16-33. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-17. QEPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	<p>Emulation mode</p> <p>0h (R/W) = QPOSCNT behavior</p> <p>Position counter stops immediately on emulation suspend</p> <p>QWDTMR behavior</p> <p>Watchdog counter stops immediately</p> <p>QUTMR behavior</p> <p>Unit timer stops immediately</p> <p>QCTMR behavior</p> <p>Capture Timer stops immediately</p> <p>1h (R/W) = QPOSCNT behavior</p> <p>Position counter continues to count until the rollover</p> <p>QWDTMR behavior</p> <p>Watchdog counter counts until WD period match roll over</p> <p>QUTMR behavior</p> <p>Unit timer counts until period rollover</p> <p>QCTMR behavior</p> <p>Capture Timer counts until next unit period event</p> <p>2h (R/W) = QPOSCNT behavior</p> <p>Position counter is unaffected by emulation suspend</p> <p>QWDTMR behavior</p> <p>Watchdog counter is unaffected by emulation suspend</p> <p>QUTMR behavior</p> <p>Unit timer is unaffected by emulation suspend</p> <p>QCTMR behavior</p> <p>Capture Timer is unaffected by emulation suspend</p> <p>3h (R/W) = Same as FREE_SOFT_2</p>
13-12	PCRM	R/W	0h	<p>Position counter reset</p> <p>0h (R/W) = Position counter reset on an index event</p> <p>1h (R/W) = Position counter reset on the maximum position</p> <p>2h (R/W) = Position counter reset on the first index event</p> <p>3h (R/W) = Position counter reset on a unit time event</p>

Table 16-17. QEPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-10	SEI	R/W	0h	Strobe event init 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
9-8	IEI	R/W	0h	Index event init of position count 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 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 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) 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 0h (R/W) = Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset. 1h (R/W) = eQEP position counter is enabled
2	QCLM	R/W	0h	QEP capture latch mode 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.

Table 16-17. QEPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	UTE	R/W	0h	QEP unit timer enable 0h (R/W) = Disable eQEP unit timer 1h (R/W) = Enable unit timer
0	WDE	R/W	0h	QEP watchdog enable 0h (R/W) = Disable the eQEP watchdog timer 1h (R/W) = Enable the eQEP watchdog timer

16.10.2.14 QCAPCTL Register (Offset = 16h) [reset = 0h]

QCAPCTL is shown in [Figure 16-34](#) and described in [Table 16-18](#).

Qaudrature Capture Control

Figure 16-34. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-18. QCAPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	CEN	R/W	0h	Enable eQEP capture 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 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 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

16.10.2.15 QPOSCTL Register (Offset = 17h) [reset = 0h]

QPOSCTL is shown in [Figure 16-35](#) and described in [Table 16-19](#).

Position Compare Control

Figure 16-35. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-19. QPOSCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	PCSHDW	R/W	0h	Position compare of shadow enable 0h (R/W) = Shadow disabled, load Immediate 1h (R/W) = Shadow enabled
14	PCLOAD	R/W	0h	Position compare of shadow load 0h (R/W) = Load on QPOSCNT = 0 1h (R/W) = Load when QPOSCNT = QPOSCMP
13	PCPOL	R/W	0h	Polarity of sync output 0h (R/W) = Active HIGH pulse output 1h (R/W) = Active LOW pulse output
12	PCE	R/W	0h	Position compare enable/disable 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 0h (R/W) = 1 * 4 * SYSCLKOUT cycles 1h (R/W) = 2 * 4 * SYSCLKOUT cycles FFFh (R/W) = 4096 * 4 * SYSCLKOUT cycles

16.10.2.16 QEINT Register (Offset = 18h) [reset = 0h]

QEINT is shown in [Figure 16-36](#) and described in [Table 16-20](#).

QEP Interrupt Control

Figure 16-36. QEINT Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-20. QEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	UTO	R/W	0h	Unit time out interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
10	IEL	R/W	0h	Index event latch interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
9	SEL	R/W	0h	Strobe event latch interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
8	PCM	R/W	0h	Position-compare match interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
7	PCR	R/W	0h	Position-compare ready interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
6	PCO	R/W	0h	Position counter overflow interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
5	PCU	R/W	0h	Position counter underflow interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
4	WTO	R/W	0h	Watchdog time out interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
3	QDC	R/W	0h	Quadrature direction change interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
2	QPE	R/W	0h	Quadrature phase error interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
1	PCE	R/W	0h	Position counter error interrupt enable 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
0	RESERVED	R	0h	Reserved

16.10.2.17 QFLG Register (Offset = 19h) [reset = 0h]

QFLG is shown in [Figure 16-37](#) and described in [Table 16-21](#).

QEP Interrupt Flag

Figure 16-37. QFLG Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-21. QFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	UTO	R	0h	Unit time out interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
10	IEL	R	0h	Index event latch interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
9	SEL	R	0h	Strobe event latch interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
8	PCM	R	0h	eQEP compare match event interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
7	PCR	R	0h	Position-compare ready interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
6	PCO	R	0h	Position counter overflow interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
5	PCU	R	0h	Position counter underflow interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
4	WTO	R	0h	Watchdog timeout interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
3	QDC	R	0h	Quadrature direction change interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
2	PHE	R	0h	Quadrature phase error interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
1	PCE	R	0h	Position counter error interrupt flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated

Table 16-21. QFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INT	R	0h	Global interrupt status flag 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated

16.10.2.18 QCLR Register (Offset = 1Ah) [reset = 0h]

QCLR is shown in [Figure 16-38](#) and described in [Table 16-22](#).

QEP Interrupt Clear

Figure 16-38. QCLR Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-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	INT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-22. QCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	UTO	R/W	0h	Clear unit time out interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
10	IEL	R/W	0h	Clear index event latch interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
9	SEL	R/W	0h	Clear strobe event latch interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
8	PCM	R/W	0h	Clear eQEP compare match event interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
7	PCR	R/W	0h	Clear position-compare ready interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
6	PCO	R/W	0h	Clear position counter overflow interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
5	PCU	R/W	0h	Clear position counter underflow interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
4	WTO	R/W	0h	Clear watchdog timeout interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
3	QDC	R/W	0h	Clear quadrature direction change interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
2	PHE	R/W	0h	Clear quadrature phase error interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
1	PCE	R/W	0h	Clear position counter error interrupt flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag

Table 16-22. QCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INT	R/W	0h	Global interrupt clear flag 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag

16.10.2.19 QFRC Register (Offset = 1Bh) [reset = 0h]

QFRC is shown in [Figure 16-39](#) and described in [Table 16-23](#).

QEP Interrupt Force

Figure 16-39. QFRC Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-23. QFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11	UTO	R/W	0h	Force unit time out interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
10	IEL	R/W	0h	Force index event latch interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
9	SEL	R/W	0h	Force strobe event latch interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
8	PCM	R/W	0h	Force position-compare match interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
7	PCR	R/W	0h	Force position-compare ready interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
6	PCO	R/W	0h	Force position counter overflow interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
5	PCU	R/W	0h	Force position counter underflow interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
4	WTO	R/W	0h	Force watchdog time out interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
3	QDC	R/W	0h	Force quadrature direction change interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
2	PHE	R/W	0h	Force quadrature phase error interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
1	PCE	R/W	0h	Force position counter error interrupt 0h (R/W) = No effect 1h (R/W) = Force the interrupt
0	RESERVED	R	0h	Reserved

16.10.2.20 QEPSTS Register (Offset = 1Ch) [reset = 8Eh]

QEPSTS is shown in [Figure 16-40](#) and described in [Table 16-24](#).

QEP Status

Figure 16-40. 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/W-1h	R-0h	R-0h	R-0h	R/W-1h	R/W-1h	R/W-1h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-24. QEPSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	UPEVNT	R/W	1h	Unit position event flag 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. 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 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 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/W	1h	Capture overflow error flag 0h (R/W) = Sticky bit, cleared by writing 1 1h (R/W) = Overflow occurred in eQEP Capture timer (QEPCTMR)
2	CDEF	R/W	1h	Capture direction error flag 0h (R/W) = Sticky bit, cleared by writing 1 1h (R/W) = Direction change occurred between the capture position event.
1	FIMF	R/W	1h	First index marker flag 0h (R/W) = Sticky bit, cleared by writing 1 1h (R/W) = Set by first occurrence of index pulse
0	PCEF	R	0h	Position counter error flag. This bit is not sticky and it is updated for every index event. 0h (R/W) = No error occurred during the last index transition 1h (R/W) = Position counter error

16.10.2.21 QCTMR Register (Offset = 1Dh) [reset = 0h]

QCTMR is shown in [Figure 16-41](#) and described in [Table 16-25](#).

QEP Capture Timer

Figure 16-41. QCTMR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCTMR															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-25. QCTMR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCTMR	R/W	0h	This register provides time base for edge capture unit.

16.10.2.22 QCPRD Register (Offset = 1Eh) [reset = 0h]

QCPRD is shown in [Figure 16-42](#) and described in [Table 16-26](#).

QEP Capture Period

Figure 16-42. QCPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCPRD															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-26. 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

16.10.2.23 QCTMRLAT Register (Offset = 1Fh) [reset = 0h]

QCTMRLAT is shown in [Figure 16-43](#) and described in [Table 16-27](#).

QEP Capture Latch

Figure 16-43. QCTMRLAT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCTMRLAT															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-27. 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.

16.10.2.24 QCPRDLAT Register (Offset = 20h) [reset = 0h]

QCPRDLAT is shown in [Figure 16-44](#) and described in [Table 16-28](#).

QEP Capture Period Latch

Figure 16-44. QCPRDLAT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCPRDLAT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 16-28. QCPRDLAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCPRDLAT	R/W	0h	eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter.

Serial Peripheral Interface (SPI)

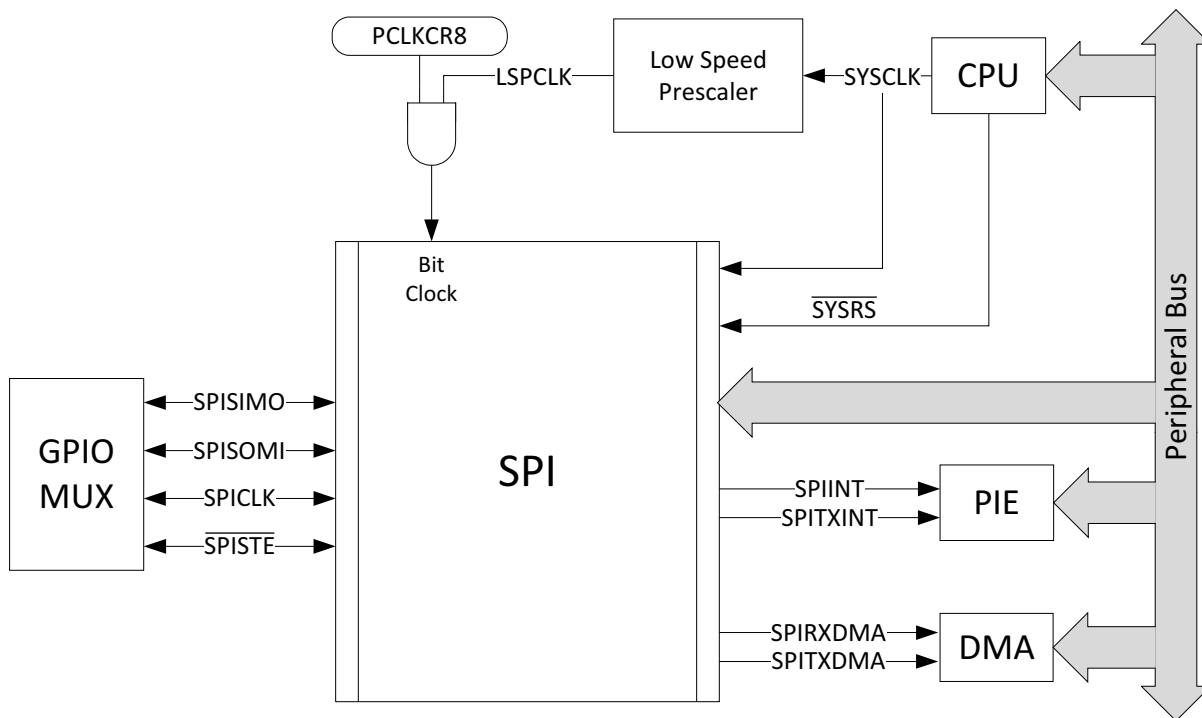
This chapter describes the serial peripheral interface (SPI) which is a high-speed synchronous serial input and output (I/O) port that allows a serial bit stream of programmed length (one to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the DSP 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
17.1 SPI Module Overview	1874
17.2 SPI Module Signal Summary	1875
17.3 Overview of SPI Module Registers	1876
17.4 Configuring Device Pins	1876
17.5 SPI Operation	1876
17.6 SPI DMA Transfers	1884
17.7 SPI Interrupts	1886
17.8 SPI FIFO Description	1887
17.9 SPI High-Speed Mode	1888
17.10 SPI 3-Wire Mode Description	1889
17.11 SPI STEINV Bit in Digital Audio Transfers	1891
17.12 SPI Waveforms	1892
17.13 Registers	1898

17.1 SPI Module Overview

Figure 17-1 shows the SPI CPU interfaces.

Figure 17-1. SPI CPU Interface



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 falling 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

- 12 SPI module control registers

NOTE: All registers in this module are 16-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7–0), and the upper byte (15–8) is read as zeros. Writing to the upper byte has no effect.

Enhanced Features:

- 16-level transmit/receive FIFO
- Delayed transmit control
- 3-wire SPI mode
- $\overline{\text{SPISTE}}$ inversion for digital audio interface receive mode on devices with two SPI modules
- DMA support
- High-speed mode for up to 40 MHz full-duplex communication

17.2 SPI Module Signal Summary

Table 17-1 classifies and provides a summary of the SPI module signals.

Table 17-1. SPI Module Signal Summary

Signal Name	Description
External Signals	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
Control	
SPI Clock Rate	LSPCLK
Interrupt Signals	
SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPI INT) Receive in interrupt in FIFO mode
SPITXINT	Transmit interrupt – FIFO
DMA Triggers	
SPITXDMA	Transmit request to DMA
SPIRXDMA	Receive request to DMA

17.3 Overview of SPI Module Registers

This SPI has 16-bit transmit and receive capability, with double-buffered transmit and double-buffered receive. All data registers are 16 bits wide.

The maximum transmission rate in both slave mode and master mode is now LSPCLK/4.

Writes of transmit data to the serial data register, SPIDAT (and the new transmit buffer, SPITXBUF), must be left-justified within a 16-bit register.

Twelve registers inside the SPI module control the SPI operations:

- SPICCR (SPI configuration control register). Contains control bits used for SPI configuration.
 - SPI module software reset
 - SPICLK polarity selection
 - Four SPI character-length control bits
 - Internal loopback selection
 - High-speed mode selection
- SPICTL (SPI operation control register). Contains control bits for data transmission.
 - Two SPI interrupt enable bits
 - SPICLK phase selection
 - Operational mode (master/slave)
 - Data transmission enable
- SPISTS (SPI status register). Contains two receive buffer status bits and one transmit buffer status bit.
 - RECEIVER OVERRUN
 - SPI INT FLAG
 - TX BUF FULL FLAG
- SPIBRR (SPI baud rate register). Contains seven bits that determine the bit transfer rate.
- SPIRXEMU (SPI receive emulation buffer register). Contains the received data. This register is used for emulation purposes only. The SPIRXBUF should be used for normal operation.
- SPIRXBUF (SPI receive buffer — the serial receive buffer register). Contains the received data.
- SPITXBUF (SPI transmit buffer — the serial transmit buffer register). Contains the next character to be transmitted.
- SPIDAT (SPI data register). Contains data to be transmitted by the SPI, acting as the transmit/receive shift register. Data written to SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI, a bit from the receive bit stream is shifted into the other end of the shift register.
- SPIPRI (SPI priority register). Contains bits that specify interrupt priority and determine SPI operation on the XDS™ emulator during program suspensions.

17.4 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.

17.5 SPI Operation

This section describes the operation of the SPI. Included are explanations of the operation modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

17.5.1 Introduction to Operation

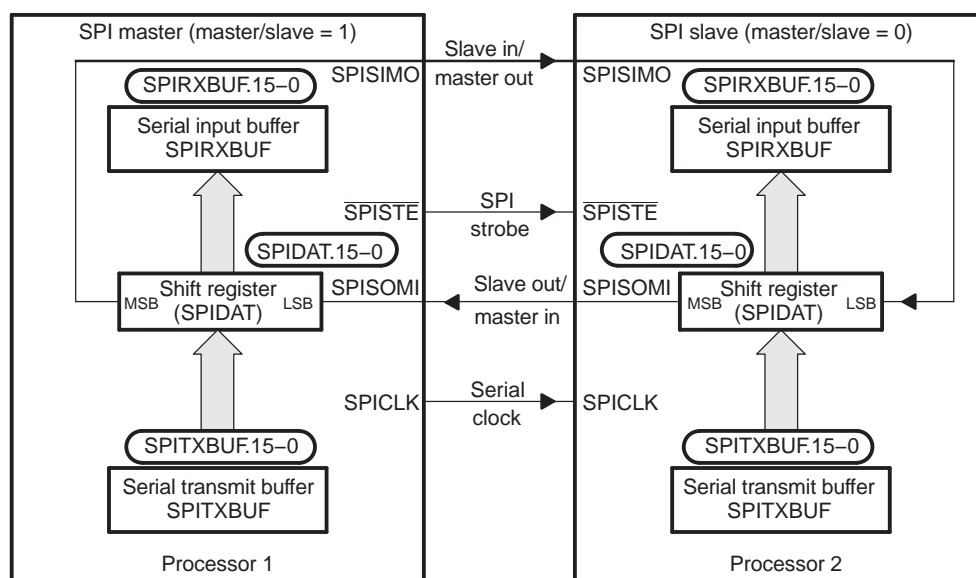
Figure 17-2 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master initiates data transfer 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 CLOCK PHASE bit (SPICLK.3) is high, data is transmitted and received a half-cycle before the SPICLK transition (see Section 17.5.2). 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 17-2. SPI Master/Slave Connection



17.5.2 SPI Module Slave and Master Operation Modes

The SPI can operate in master or slave mode. The MASTER/SLAVE bit (SPICLK.2) selects the operating mode and the source of the SPICLK signal.

17.5.2.1 Master Mode

In the 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 126 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.

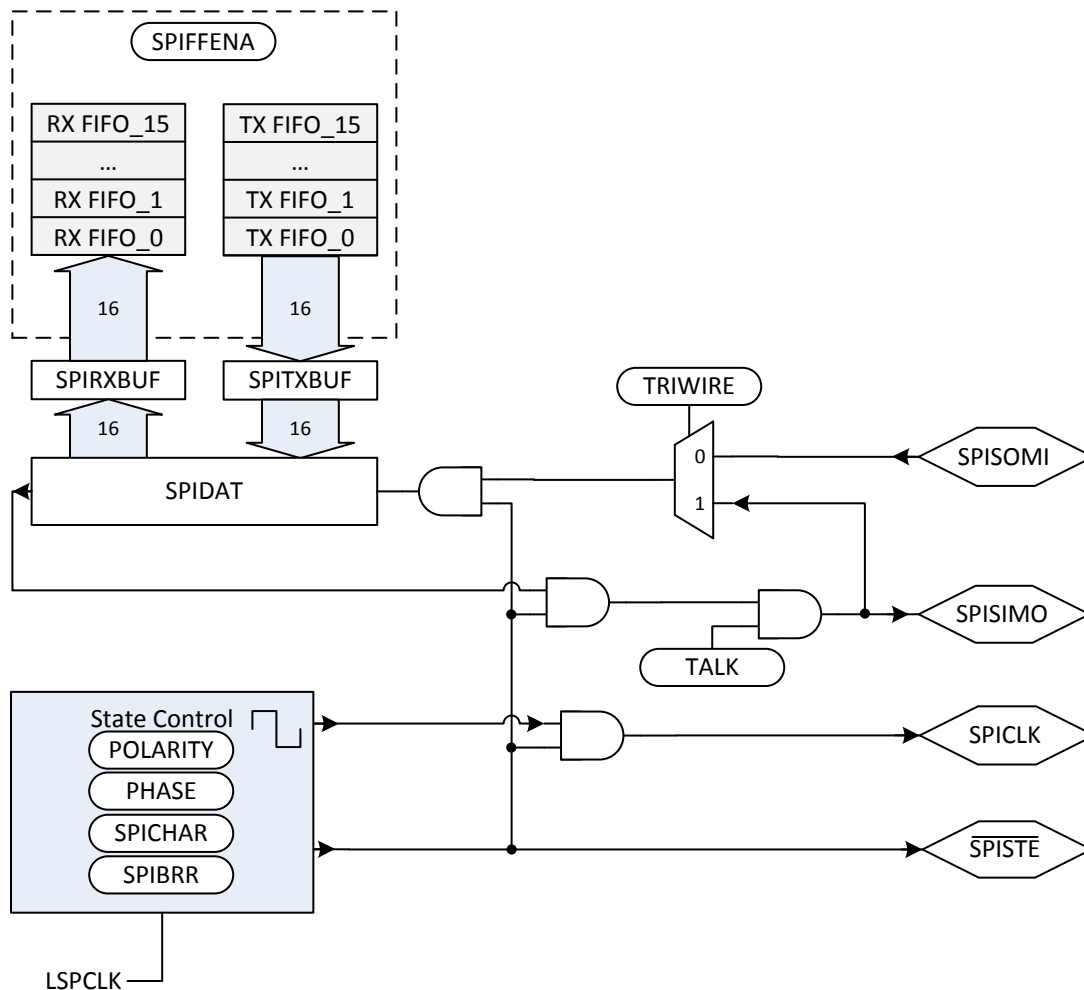
When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- SPIDAT contents are transferred to SPIRXBUF.
- SPI INT FLAG bit (SPISTS.6) is set to 1.
- If there is valid data in the transmit buffer SPITXBUF, as indicated by the TXBUF FULL bit in SPISTS, this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- If the SPI INT ENA bit (SPICTL.0) 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 17-3 is a block diagram of the SPI in master mode. It shows the basic control blocks available in SPI master mode.

Figure 17-3. SPI Module Master Configuration



17.5.2.2 Slave Mode

In the slave mode (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.

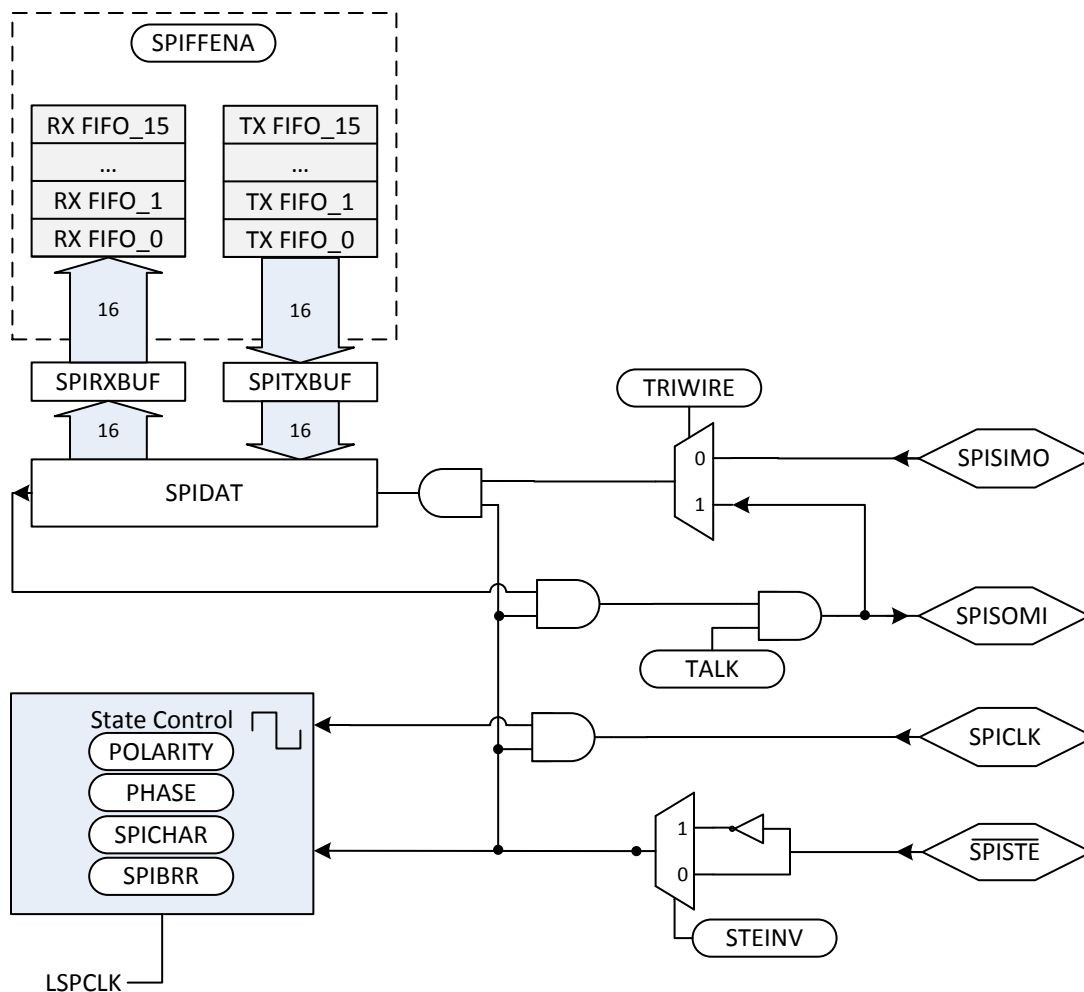
Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of 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 (SPICTL.1) 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 17-4 is a block diagram of the SPI in slave mode. It shows the basic control blocks available in SPI slave mode.

Figure 17-4. SPI Module Slave Configuration



17.5.3 Initialization Upon Reset

A system reset forces the SPI peripheral module 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
- SPI module pin functions are selected as general-purpose inputs (this is done in I/O MUX control register B [MCRB])

To change this SPI configuration:

- Step 1. Clear the SPI SW RESET bit (SPICCR.7) to 0 to force the SPI to the reset state.
- Step 2. Initialize the SPI configuration, format, baud rate, and pin functions as desired.
- Step 3. Set the SPI SW RESET bit to 1 to release the SPI from the reset state.
- Step 4. Write to SPIDAT or SPITXBUF (this initiates the communication process in the master).
- Step 5. Read SPIRXBUF after the data transmission has completed (SPISTS.6 = 1) to determine what data was received.

To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPI SW RESET bit (SPICCR.7) before making initialization changes, and then set this bit after initialization is complete.

NOTE: Do not change the SPI configuration when communication is in progress.

17.5.4 Data Format

Four bits (SPICCR.3–0) specify the number of bits (1 to 16) in the data character. 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 17-1](#)).

Example 17-1. Transmission of Bit From SPIRXBUF

Conditions:

1. Transmission character length = 1 bit (specified in bits SPICCR.3–0)
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
SPIDAT (after transmission)															
(TXed) 0 ←	1	1	1	0	0	1	1	0	1	1	1	1	0	1	x ⁽¹⁾
SPIRXBUF (after transmission)															
	1	1	1	0	0	1	1	0	1	1	1	1	0	1	x ⁽¹⁾

⁽¹⁾ x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

17.5.5 Baud Rate and Clocking Schemes

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.

Example 17-2 shows how to determine the SPI baud rates.

Example 17-2. Baud Rate Determination

For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)} \quad (5)$$

For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4} \quad (6)$$

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 17-3. Baud Rate Calculation in Non-High Speed Mode (HS_MODE=0)

$$\begin{aligned} \text{SPI Baud Rate} &= \frac{\text{LSPCLK}}{\text{SPIBRR} + 1}, \quad \text{LSPCLK} = 50 \text{ MHz} \\ &= \frac{50 \times 10^6}{3 + 1} \\ &= 12.5 \text{ Mbps} \end{aligned} \quad (7)$$

17.5.5.1 SPI Clocking Schemes

The CLOCK POLARITY bit (SPICCR.6) and the CLOCK PHASE bit (SPICTL.3) control four different clocking schemes on the SPICLK pin. The CLOCK POLARITY bit selects the active edge, either rising or falling, of the clock. The CLOCK PHASE bit 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.

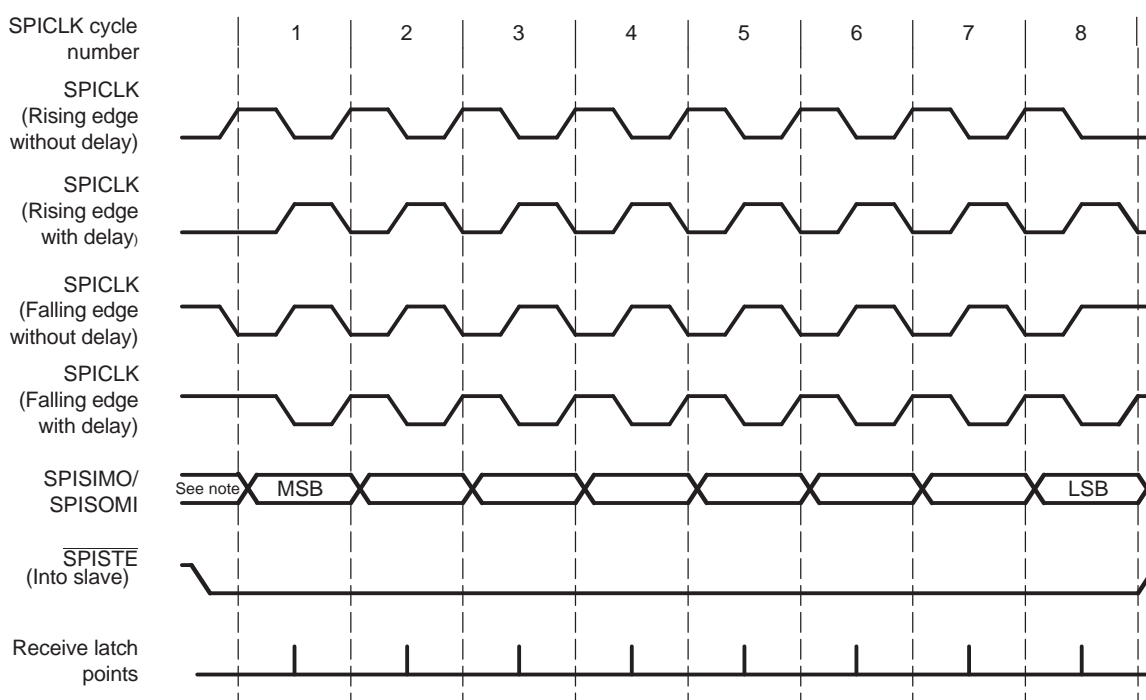
The selection procedure for the SPI clocking scheme is shown in [Table 17-2](#). Examples of these four clocking schemes relative to transmitted and received data are shown in [Figure 17-5](#).

Table 17-2. SPI Clocking Scheme Selection Guide

SPICLK Scheme	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3) ⁽¹⁾
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

⁽¹⁾ The description of CLOCK PHASE and CLOCK POLARITY differs between manufacturers. For proper operation, select the desired waveform to determine the PHASE and POLARITY settings.

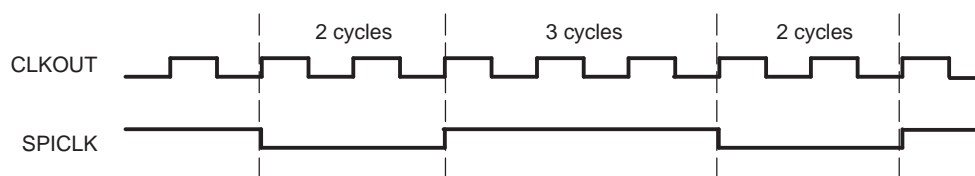
Figure 17-5. SPICLK Signal Options



Note: Previous data bit

For the SPI, SPICLK symmetry is retained only when the result of (SPIBRR+1) is an even value. When (SPIBRR + 1) is an odd value and SPIBRR is greater than 3, SPICLK becomes asymmetrical. The low pulse of SPICLK is one CLKOUT longer than the high pulse when the CLOCK POLARITY bit is clear (0). When the CLOCK POLARITY bit is set to 1, the high pulse of the SPICLK is one CLKOUT longer than the low pulse, as shown in [Figure 17-6](#).

Figure 17-6. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1

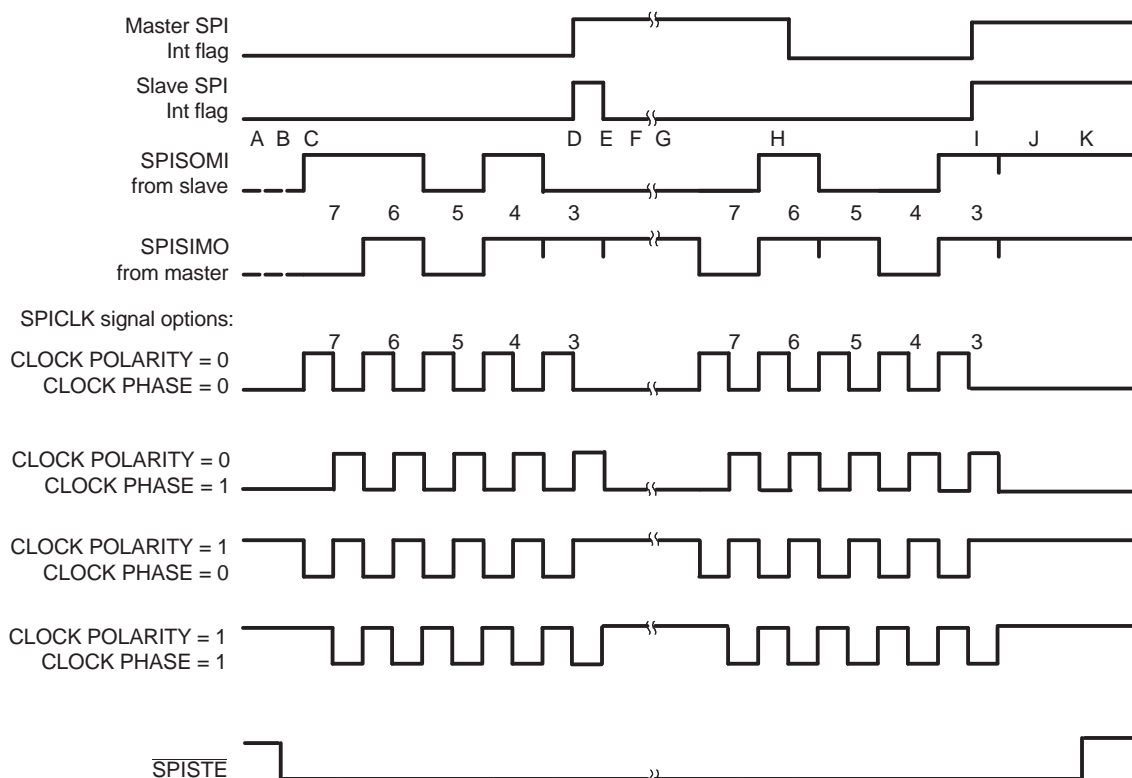


17.5.6 Data Transfer Example

The timing diagram shown in [Figure 17-7](#) 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 unsymmetrical ([Figure 17-6](#)) shares similar characterizations with [Figure 17-7](#) except that the data transfer is one CLKOUT cycle longer per bit during the low pulse (CLOCK POLARITY = 0) or during the high pulse (CLOCK POLARITY = 1) of the SPICLK.

[Figure 17-7](#) is applicable for 8-bit SPI only and is not for 28x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.

Figure 17-7. 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).

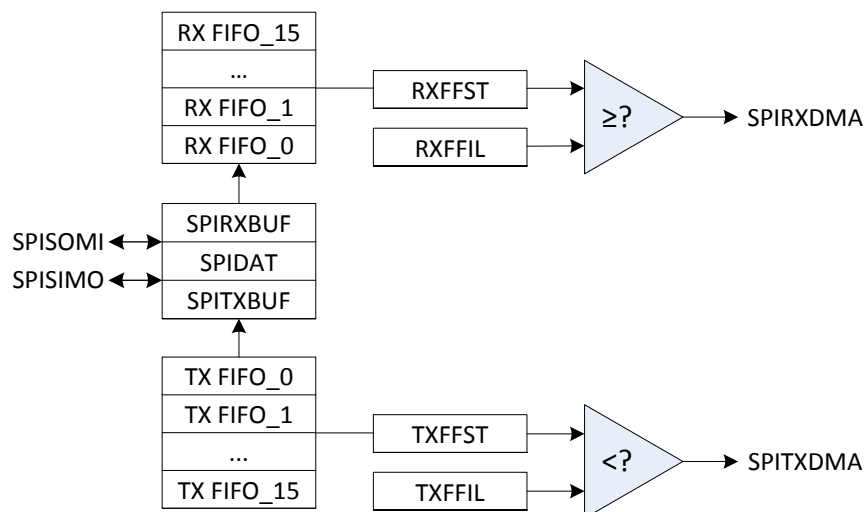
17.6 SPI DMA Transfers

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 user should ensure the following conditions for proper data transfers.

NOTE: In order to use the DMA with SPI, the FIFOs must be enabled.

Figure 17-8 is a block diagram showing the DMA trigger generation from the SPI module.

Figure 17-8. SPI DMA Trigger Diagram



17.6.1 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 – 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 of 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)

17.6.2 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)

17.7 SPI Interrupts

This section includes information on the control bits that initialize interrupts, data format, clocking, initialization, and data transfer. Figure 17-9 and Table 17-3 show how these control bits influence the SPI interrupt generation.

Figure 17-9. SPI Interrupt Flags and Enable Logic Generation

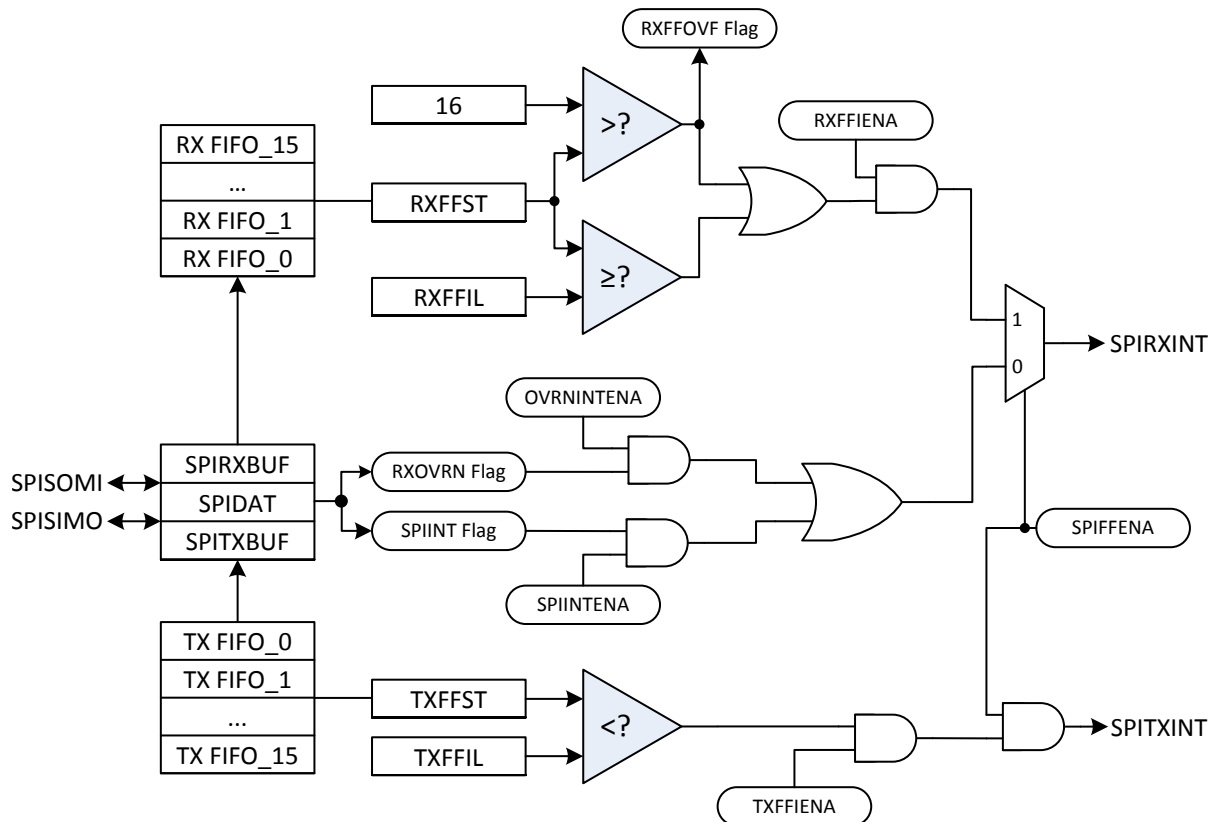


Table 17-3. SPI Interrupt Flag Modes

FIFO Options	SPI interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable (SPIFFENA)	Interrupt ⁽¹⁾ 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

⁽¹⁾ In non-FIFO mode, SPIRXINT is the same name as the SPIINT interrupt in 28x devices.

17.7.1 SPI Interrupt Control Bits

Five control bits are used to initialize the SPI interrupt:

- SPI INT ENA bit (SPICTL.0)
- SPI INT FLAG bit (SPISTS.6)
- OVERRUN INT ENA bit (SPICTL.4)
- RECEIVER OVERRUN FLAG bit (SPISTS.7)

17.7.1.1 SPI INT ENA Bit (SPICTL.0)

When the SPI interrupt-enable bit is set and an interrupt condition occurs, the corresponding interrupt is asserted.

- | | |
|---|------------------------|
| 0 | Disable SPI interrupts |
| 1 | Enable SPI interrupts |

17.7.1.2 SPI INT FLAG Bit (SPISTS.6)

This status flag indicates that a character has been placed in the SPI receiver buffer and is ready to be read.

When a complete character has been shifted into or out of SPIDAT, the SPI INT FLAG bit (SPISTS.6) is set, and an interrupt is generated if enabled by the SPI INT ENA bit (SPICTL.0). The interrupt flag remains set until it is cleared by one of the following events:

- The interrupt is acknowledged.
- The CPU reads the SPIRXBUF (reading the SPIRXEMU does not clear the SPI INT FLAG bit).
- The device enters IDLE2 or HALT mode with an IDLE instruction.
- Software clears the SPI SW RESET bit (SPICCR.7).
- A system reset occurs.

When the SPI INT FLAG bit is set, a character has been placed into the SPIRXBUF and is ready to be read. If the CPU does not read the character by the time the next complete character has been received, the new character is written into SPIRXBUF, and the RECEIVER OVERRUN Flag bit (SPISTS.7) is set.

17.7.1.3 OVERRUN INT ENA Bit (SPICTL.4)

Setting the overrun interrupt enable bit allows the assertion of an interrupt whenever the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by SPISTS.7 and by the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.

- | | |
|---|--|
| 0 | Disable RECEIVER OVERRUN Flag bit interrupts |
| 1 | Enable RECEIVER OVERRUN Flag bit interrupts |

17.7.1.4 RECEIVER OVERRUN FLAG Bit (SPISTS.7)

The RECEIVER OVERRUN Flag bit is set whenever a new character is received and loaded into the SPIRXBUF before the previously received character has been read from the SPIRXBUF. The RECEIVER OVERRUN Flag bit must be cleared by software.

17.8 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, 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 SPIFFEN 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 transmit FIFO, SPITXINT and one for receive FIFO, SPIINT/SPIRXINT. SPIINT/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.
6. **Buffers.** Transmit and receive buffers are supplemented with two 16x16 FIFOs. The one-word transmit buffer (TXBUF) 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 serial clock cycles and a maximum of 255 serial clock 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 SPI clocks 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 (bits 12– 0) 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 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 SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

17.9 SPI High-Speed Mode

The SPI module is capable of reaching full-duplex communication speeds up to 40 MHz.

In order to achieve the maximum full-duplex speeds, two restrictions are placed on the design:

- High-speed capability is available on a single pin mux option
- Single master to single slave configuration is supported to limit the loading on the pins.

17.9.1 GPIOs Required for High-Speed Mode

The high-speed mode of the SPI is only available on specific GPIO mux options. To enable the high-speed enhancements, set SPICCR.HS_MODE to 1.

When not operating in high-speed mode, SPICCR.HS_MODE should be set to 0. The high-speed capable GPIOs may still be used as standard SPI pins if HS_MODE is not enabled. They are explained in [Table 17-4](#).

Table 17-4. High-Speed SPI Capable GPIOs

SPI pin	SPIA	SPIB	SPIC
SPISIMO	GPIO58	GPIO63	GPIO69
SPISOMI	GPIO59	GPIO64	GPIO70
SPICLK	GPIO60	GPIO65	GPIO71
SPISTE	GPIO61	GPIO66	GPIO72

17.9.2 Configuring the SPI for High-Speed Mode

In order to achieve 40 MHz full-duplex operation, the following settings must be made. This example assumes that the device is operating at 200 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 SPICCR.HS_MODE to 1.

```
SpiaRegs.SPICCR.bit.HS_MODE = 0x1;
```

Set SPIBRR to 4. $SPICLK = LSPCLK / (SPIBRR + 1) = 40 \text{ MHz}$

```
SpiaRegs.SPIBRR = 0x4;
```

There are no other differences in the configuration from normal SPI mode. Sending and receiving data is also the same.

17.10 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 (SPIPRI.0) 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 17-5](#) indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

Table 17-5. 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 (SPICTL.1) 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 17-10](#) and [Figure 17-11](#) illustrate 3-wire master and slave mode.

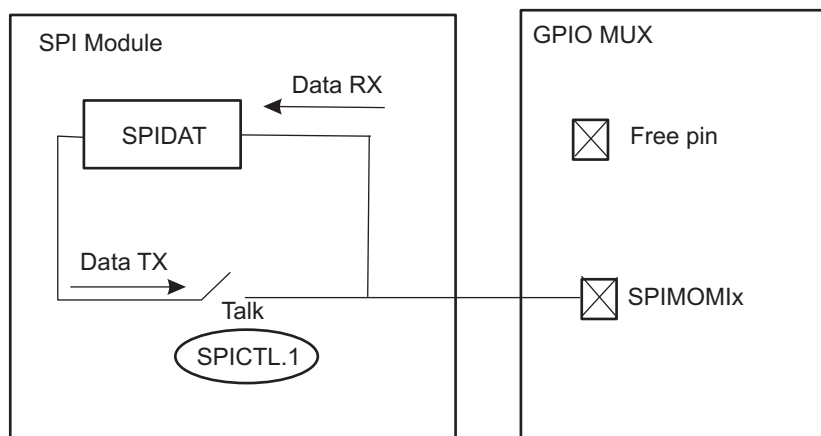
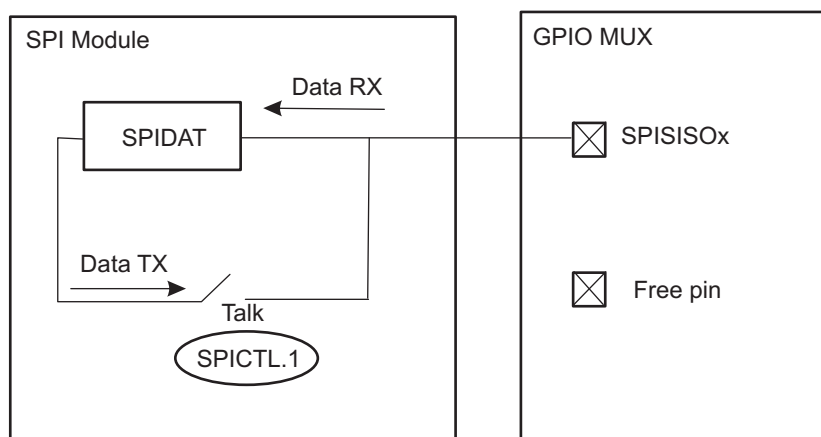
Figure 17-10. SPI 3-wire Master Mode

Figure 17-11. SPI 3-wire Slave Mode


Table 17-6 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

Table 17-6. 3-Wire SPI Pin Configuration

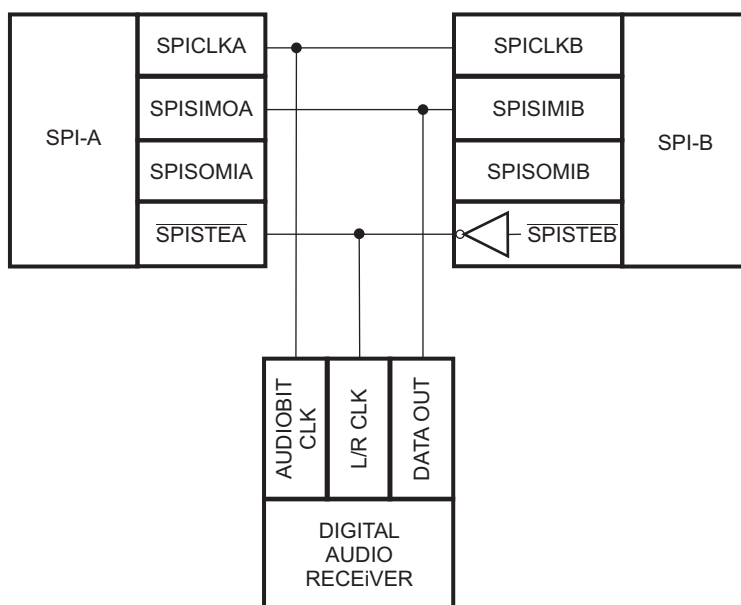
Pin Mode	SPIPRI[TRIWIRE]	SPICTL[TALK]	SPISIMO	SPISOMI
Master Mode				
4-wire	0	X	TX	RX
3-pin mode	1	0	RX	Disconnect from SPI
		1	TX/RX	
Slave Mode				
4-wire	0	X	RX	TX
3-pin mode	1	0	Disconnect from SPI	RX
		1		TX/RX

17.11 SPI STEINV Bit in Digital Audio Transfers

On those devices with two SPI modules, enabling the STEINV bit (SPIPRI.1) 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 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 17-12.

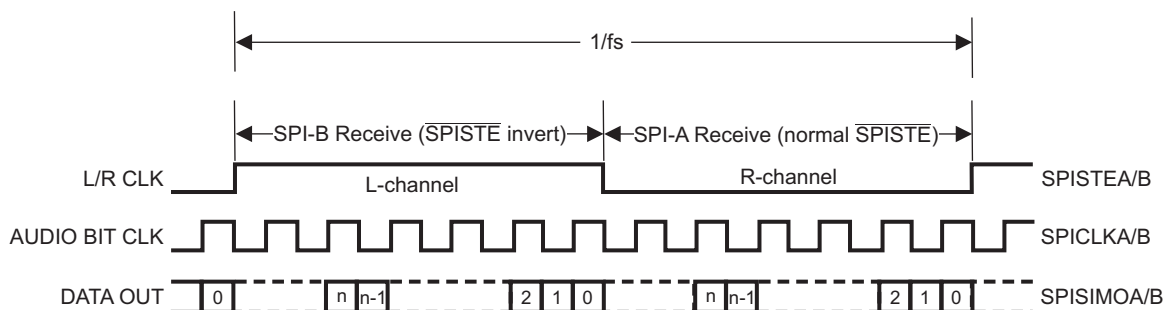
NOTE: This configuration is only applicable to slave mode (MASTER_SLAVE = 0). When the SPI is configured as master (MASTER_SLAVE = 1), the STEINV bit will have no effect on the $\overline{\text{SPISTE}}$ pin.

Figure 17-12. SPI Digital Audio Receiver Configuration Using 2 SPIs



Standard 28x 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 electricals for SPI timing requirements. With the SPI clock phase configured such that the CLOCK POLARITY (SPICCR.6) bit is 0 and the CLOCK PHASE (SPICTL.3) bit is 1 (data latched on rising edge of clock), standard right-justified digital audio interface data format is supported as shown in Figure 17-13.

Figure 17-13. Standard Right-Justified Digital Audio Data Format



17.12 SPI Waveforms

The SPI waveforms are shown in [Figure 17-14](#) through [Figure 17-19](#).

Figure 17-14. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)

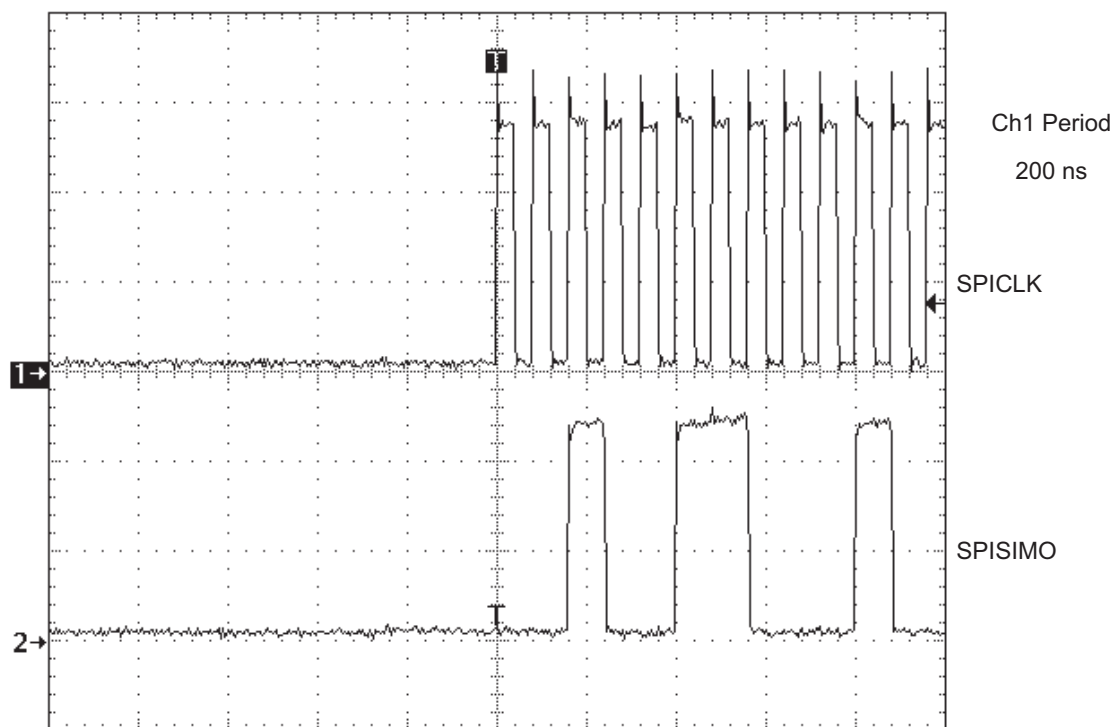


Figure 17-15. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.)

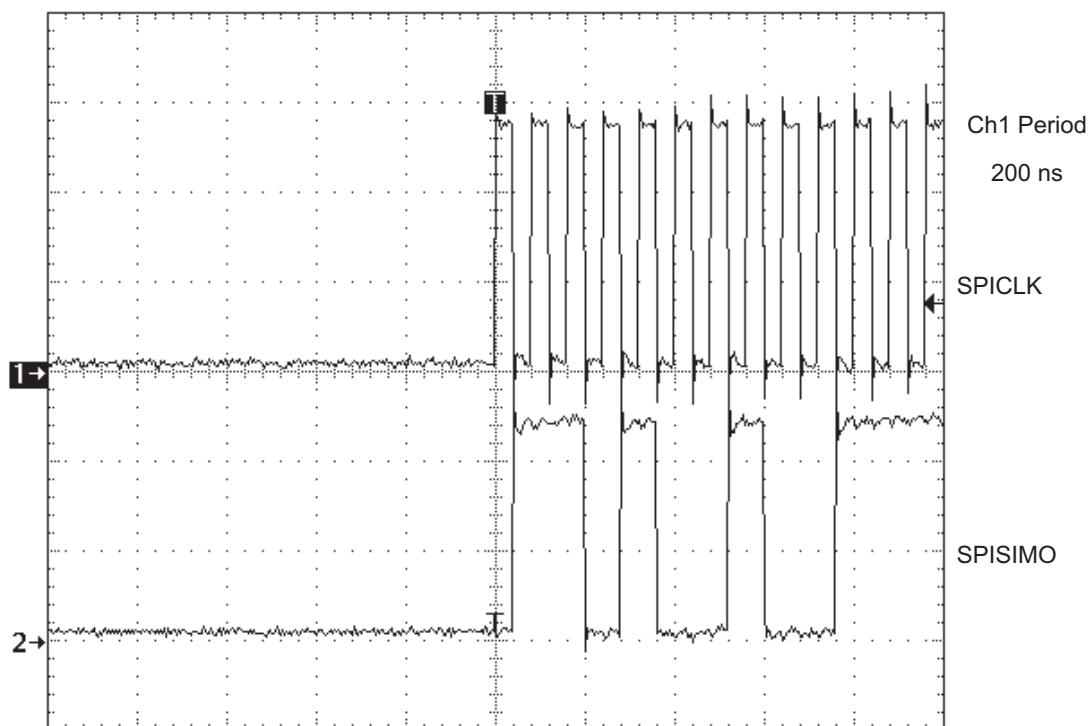


Figure 17-16. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.)

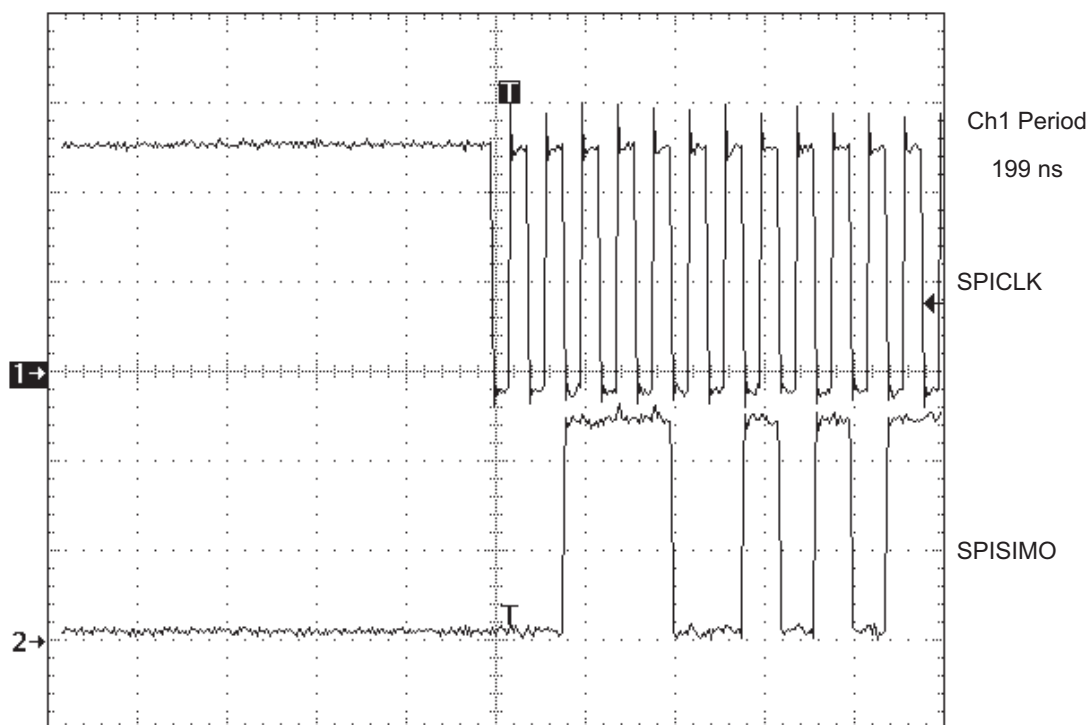


Figure 17-17. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)

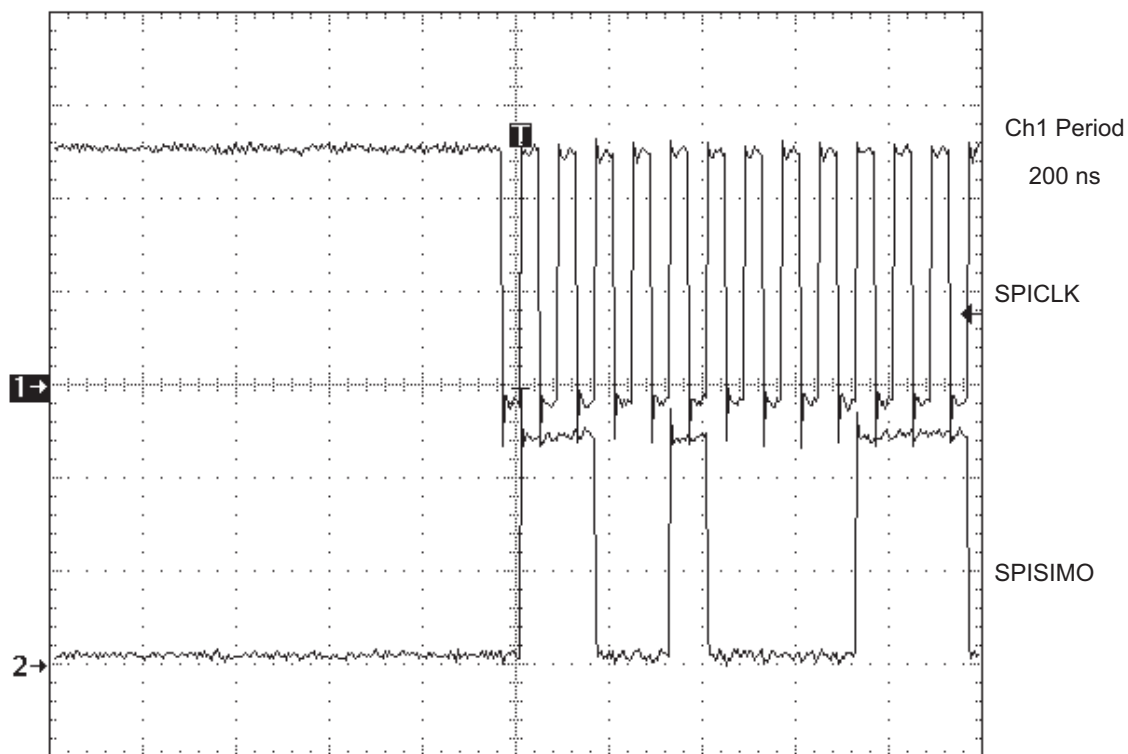


Figure 17-18. $\overline{\text{SPISTE}}$ Behavior in Master Mode (Master lowers $\overline{\text{SPISTE}}$ during the entire 16 bits of transmission.)

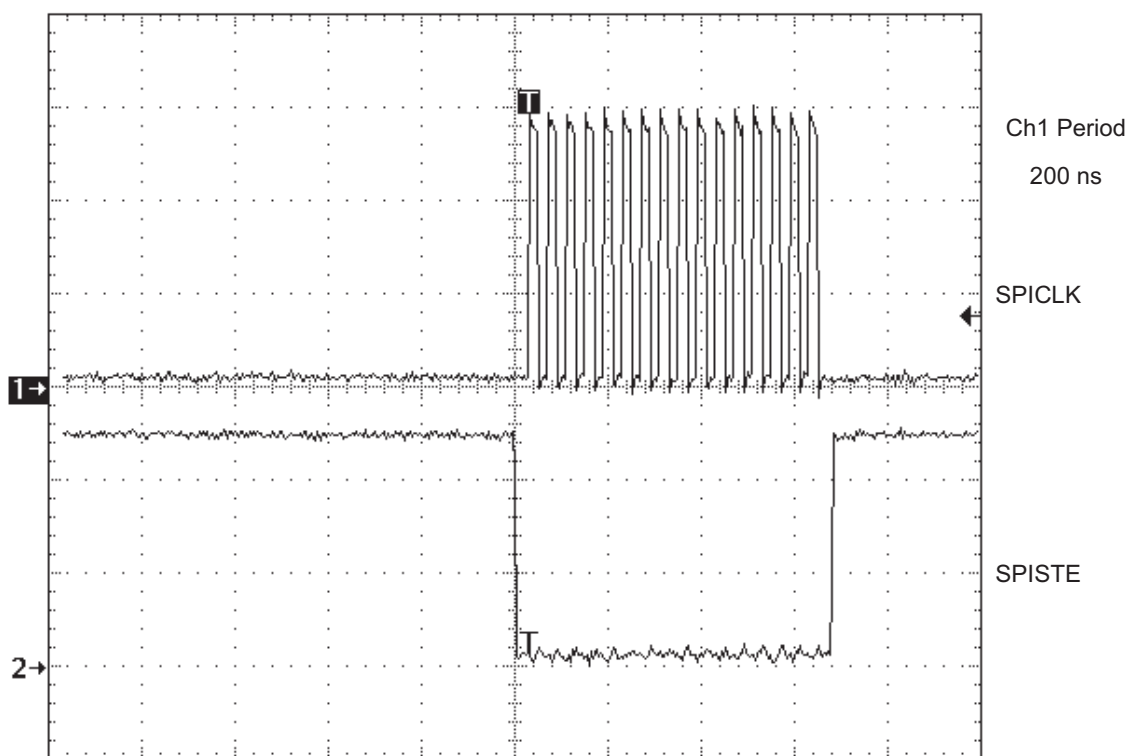
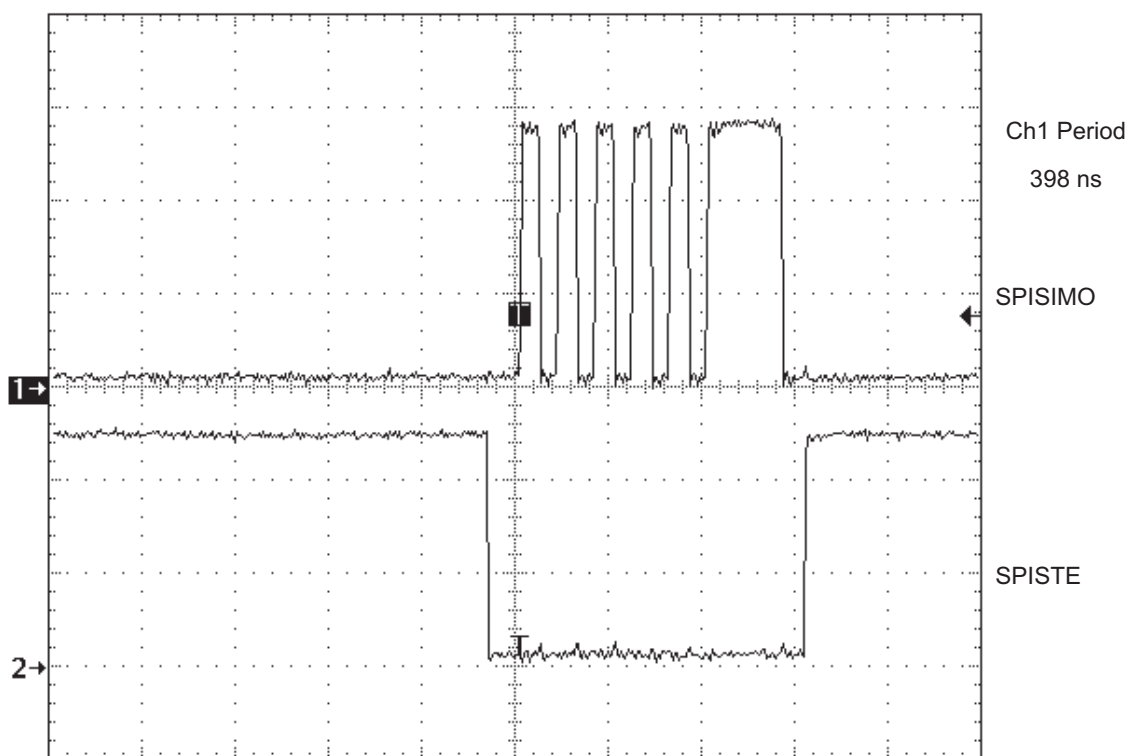


Figure 17-19. $\overline{\text{SPISTE}}$ Behavior in Slave Mode (Slave's $\overline{\text{SPISTE}}$ is lowered during the entire 16 bits of transmission.)



17.13 Registers

17.13.1 SPI Base Addresses

The base addresses for SPI are shown in [Table 17-7](#).

Table 17-7. 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
SpicRegs	SPI_REGS	0x0000_6120	0x0000_612F

17.13.2 SPI_REGS Registers

Table 17-8 lists the memory-mapped registers for the SPI_REGS. All register offset addresses not listed in Table 17-8 should be considered as reserved locations and the register contents should not be modified.

Table 17-8. SPI_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SPICCR	SPI Configuration Control Register		Go
1h	SPICTL	SPI Operation Control Register		Go
2h	SPISTS	SPI Status Register		Go
4h	SPIBRR	SPI Baud Rate Register		Go
6h	SPIRXEMU	SPI Emulation Buffer Register		Go
7h	SPIRXBUF	SPI Serial Input Buffer Register		Go
8h	SPITXBUF	SPI Serial Output Buffer Register		Go
9h	SPIDAT	SPI Serial Data Register		Go
Ah	SPIFFTX	SPI FIFO Transmit Register		Go
Bh	SPIFFRX	SPI FIFO Receive Register		Go
Ch	SPIFFCT	SPI FIFO Control Register		Go
Fh	SPIPRI	SPI Priority Control Register		Go

17.13.2.1 SPICCR Register (Offset = 0h) [reset = 0h]

SPICCR is shown in [Figure 17-20](#) and described in [Table 17-9](#).

SPI Configuration Control Register

Figure 17-20. 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-0h	R/W-0h	R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-9. 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>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. 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.</p>
6	CLKPOLARITY	R	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>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"> - 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. - 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. <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"> - 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. - 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.

Table 17-9. SPICCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	HS_MODE	R/W	0h	High Speed Mode Enable Bits This bit determines if the High Speed mode is enabled. The correct GPIOs should be selected in the GPxGMUX/GPxMUX registers. 0h (R/W) = SPI High Speed mode disabled. This is the default value after reset. 1h (R/W) = SPI High Speed mode enabled,
4	SPILBK	R/W	0h	SPI Loopback Mode Select Loopback mode allows module validation during device testing. This mode is valid only in master mode of the SPI. 0h (R/W) = SPI loopback mode disabled. This is the default value after reset. 1h (R/W) = SPI loopback mode enabled, SIMO/SOMI lines are connected internally. Used for module self-tests.
3-0	SPICHR	R/W	0h	Character Length Control Bits These four bits determine the number of bits to be shifted in or SPI CHAR0 out as a single character during one shift sequence.

17.13.2.2 SPICTL Register (Offset = 1h) [reset = 0h]

SPICTL is shown in [Figure 17-21](#) and described in [Table 17-10](#).

SPI Operation Control Register

Figure 17-21. 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_SLAVE	TALK	SPIINTENA
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-10. 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>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>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>0h (R/W) = SPI is configured as a slave.</p> <p>1h (R/W) = SPI is configured as a master.</p>

Table 17-10. 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>0h (R/W) = Disables transmission:</p> <ul style="list-style-type: none"> - 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. - 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. <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>0h (R/W) = Disables the interrupt.</p> <p>1h (R/W) = Enables the interrupt.</p>

17.13.2.3 SPISTS Register (Offset = 2h) [reset = 0h]

SPISTS is shown in [Figure 17-22](#) and described in [Table 17-11](#).

SPI Status Register

Figure 17-22. 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 AG	RESERVED				
W1toClr-0h	RtoClr-0h	RtoClr-0h	R-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-11. SPISTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	OVERRUN_FLAG	W1toClr	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"> - Writing a 1 to this bit - Writing a 0 to SPI SW RESET (SPICCR.7) - Resetting the system <p>If the OVERRUN INT ENA bit (SPICCTL.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>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>
6	INT_FLAG	RtoClr	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 (SPICCTL.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"> - Reading SPIRXBUF - Writing a 0 to SPI SW RESET (SPICCR.7) - Resetting the system <p>0h (R/W) = No full words have been received or transmitted.</p> <p>1h (R/W) = Indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced.</p>

Table 17-11. SPISTS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	BUFFULL_FLAG	RtoClr	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>0h (R/W) = Transmit buffer is not full. 1h (R/W) = Transmit buffer is full.</p>
4-0	RESERVED	R	0h	Reserved

17.13.2.4 SPIBRR Register (Offset = 4h) [reset = 0h]

SPIBRR is shown in [Figure 17-23](#) and described in [Table 17-12](#).

SPI Baud Rate Register

Figure 17-23. 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						

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-12. 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 SPICLK 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>where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device.</p> <p>3h (R/W) = SPI Baud Rate = LSPCLK/4 4h (R/W) = SPI Baud Rate = LSPCLK/5 7Eh (R/W) = SPI Baud Rate = LSPCLK/127 7Fh (R/W) = SPI Baud Rate = LSPCLK/128</p>

17.13.2.5 SPIRXEMU Register (Offset = 6h) [reset = 0h]

SPIRXEMU is shown in [Figure 17-24](#) and described in [Table 17-13](#).

SPI Emulation Buffer Register

Figure 17-24. SPIRXEMU Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERXBn															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-13. 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>

17.13.2.6 SPIRXBUF Register (Offset = 7h) [reset = 0h]

SPIRXBUF is shown in [Figure 17-25](#) and described in [Table 17-14](#).

SPI Serial Input Buffer Register

Figure 17-25. SPIRXBUF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBn															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-14. 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>

17.13.2.7 SPITXBUF Register (Offset = 8h) [reset = 0h]

SPITXBUF is shown in [Figure 17-26](#) and described in [Table 17-15](#).

SPI Serial Output Buffer Register

Figure 17-26. SPITXBUF Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXBn															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-15. SPITXBUF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TXBn	R/W	0h	<p>Transmit Data Buffer</p> <p>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.</p>

17.13.2.8 SPIDAT Register (Offset = 9h) [reset = 0h]

SPIDAT is shown in [Figure 17-27](#) and described in [Table 17-16](#).

SPI Serial Data Register

Figure 17-27. SPIDAT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDATn															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-16. SPIDAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	SDATn	R	0h	<p>Serial Data Shift Register</p> <ul style="list-style-type: none"> - It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set. - When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, see 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. <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>

17.13.2.9 SPIFFTX Register (Offset = Ah) [reset = A000h]

SPIFFTX is shown in [Figure 17-28](#) and described in [Table 17-17](#).

SPI FIFO Transmit Register

Figure 17-28. 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/W-0h	W-0h	R/W-0h			R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-17. SPIFFTX Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SPIRST	R/W	1h	SPI Reset 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 0h (R/W) = SPI FIFO enhancements are disabled. 1h (R/W) = SPI FIFO enhancements are enabled.
13	TXFIFO	R/W	1h	TX FIFO Reset 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 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/W	0h	TX FIFO Interrupt Flag 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 0h (R/W) = Write 0 has no effect on TXFIFINT 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 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 17-17. 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>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 word or no words 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>

17.13.2.10 SPIFFRX Register (Offset = Bh) [reset = A000h]

SPIFFRX is shown in [Figure 17-29](#) and described in [Table 17-18](#).

SPI FIFO Receive Register

Figure 17-29. SPIFFRX Register

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVFCLR	RXFIFORESET	RXFFST				
R/W-1h	R/W-0h	R/W-1h	R-0h				
7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL				
R/W-0h	W-0h	R/W-0h	R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-18. SPIFFRX Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RXFFOVF	R/W	1h	Receive FIFO Overflow Flag 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	R/W	0h	Receive FIFO Overflow Clear 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 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 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/W	0h	Receive FIFO Interrupt Flag 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 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
5	RXFFIENA	R/W	0h	RX FIFO Interrupt Enable 0h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be disabled. 1h (R/W) = RX FIFO interrupt based on RXFFIL match (greater than or equal to) will be enabled.

Table 17-18. SPIFFRX Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-0	RXFFIL	R/W	0h	<p>Receive FIFO Interrupt Level Bits</p> <p>11111 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>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>

17.13.2.11 SPIFFCT Register (Offset = Ch) [reset = 0h]

SPIFFCT is shown in [Figure 17-30](#) and described in [Table 17-19](#).

SPI FIFO Control Register

Figure 17-30. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-19. 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>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>

17.13.2.12 SPIPRI Register (Offset = Fh) [reset = 0h]

SPIPRI is shown in [Figure 17-31](#) and described in [Table 17-20](#).

SPI Priority Control Register

Figure 17-31. SPIPRI Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	PRIORITY	SOFT	FREE	RESERVED		STEINV	TRIWIRES
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h		R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 17-20. SPIPRI Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	PRIORITY	R/W	0h	Interrupt Priority Select
5	SOFT	R/W	0h	Emulation Soft Run This bit only has an effect when the FREE bit is 0. 0h (R/W) = Transmission stops after 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. The SCI module operates differently. 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. Standard SPI mode: Stop after transmitting the words in the shift register and buffer. That is, after TXBUF and SPIDAT are empty. In FIFO mode: Stop after transmitting the words in the shift register and buffer. That is, after TX FIFO and SPIDAT are empty.
4	FREE	R/W	0h	Emulation Free Run 0h (R/W) = Emulation mode is selected by the SOFT bit 1h (R/W) = Free run, continue SPI operation regardless of suspend or when the suspend occurred.
3-2	RESERVED	R	0h	Reserved
1	STEINV	R/W	0h	SPISTEN Inversion Bit 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. This bit is only applicable to slave mode. Writing to this bit while configured as master (MASTER_SLAVE = 1) has no effect 0h (R/W) = SPISTEN is active low (normal) 1h (R/W) = SPISTE is active high (inverted)
0	TRIWIRES	R/W	0h	SPI 3-wire Mode Enable 0h (R/W) = Normal 4-wire SPI mode. 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.

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.

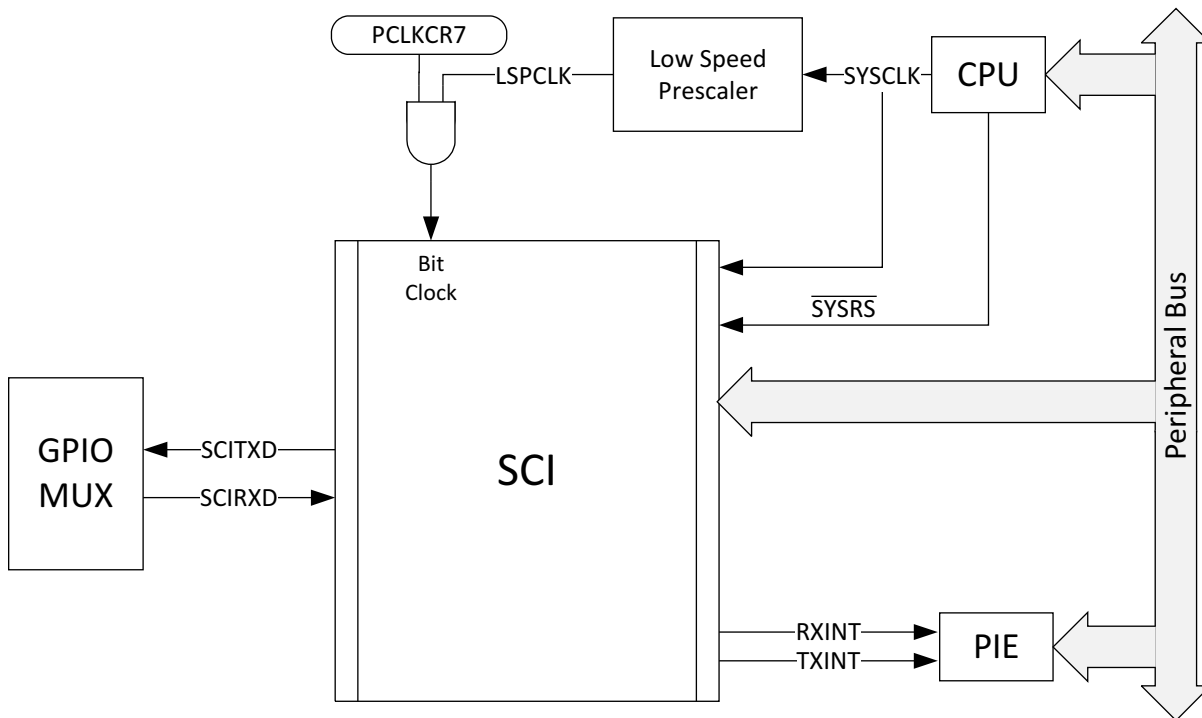
To specify data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to different speeds through a 16-bit baud-select register.

Topic	Page
18.1 Enhanced SCI Module Overview	1918
18.2 Architecture	1919
18.3 SCI Module Signal Summary	1920
18.4 Configuring Device Pins	1920
18.5 Multiprocessor and Asynchronous Communication Modes	1920
18.6 SCI Programmable Data Format.....	1921
18.7 SCI Multiprocessor Communication	1921
18.8 Idle-Line Multiprocessor Mode	1922
18.9 Address-Bit Multiprocessor Mode	1924
18.10 SCI Communication Format	1925
18.11 SCI Port Interrupts	1927
18.12 SCI Baud Rate Calculations	1927
18.13 SCI Enhanced Features	1928
18.14 Registers	1931

18.1 Enhanced SCI Module Overview

The SCI interfaces are shown in [Figure 18-1](#).

Figure 18-1. 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
- 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
- 13 SCI module control registers located in the control register frame beginning at address 7050h

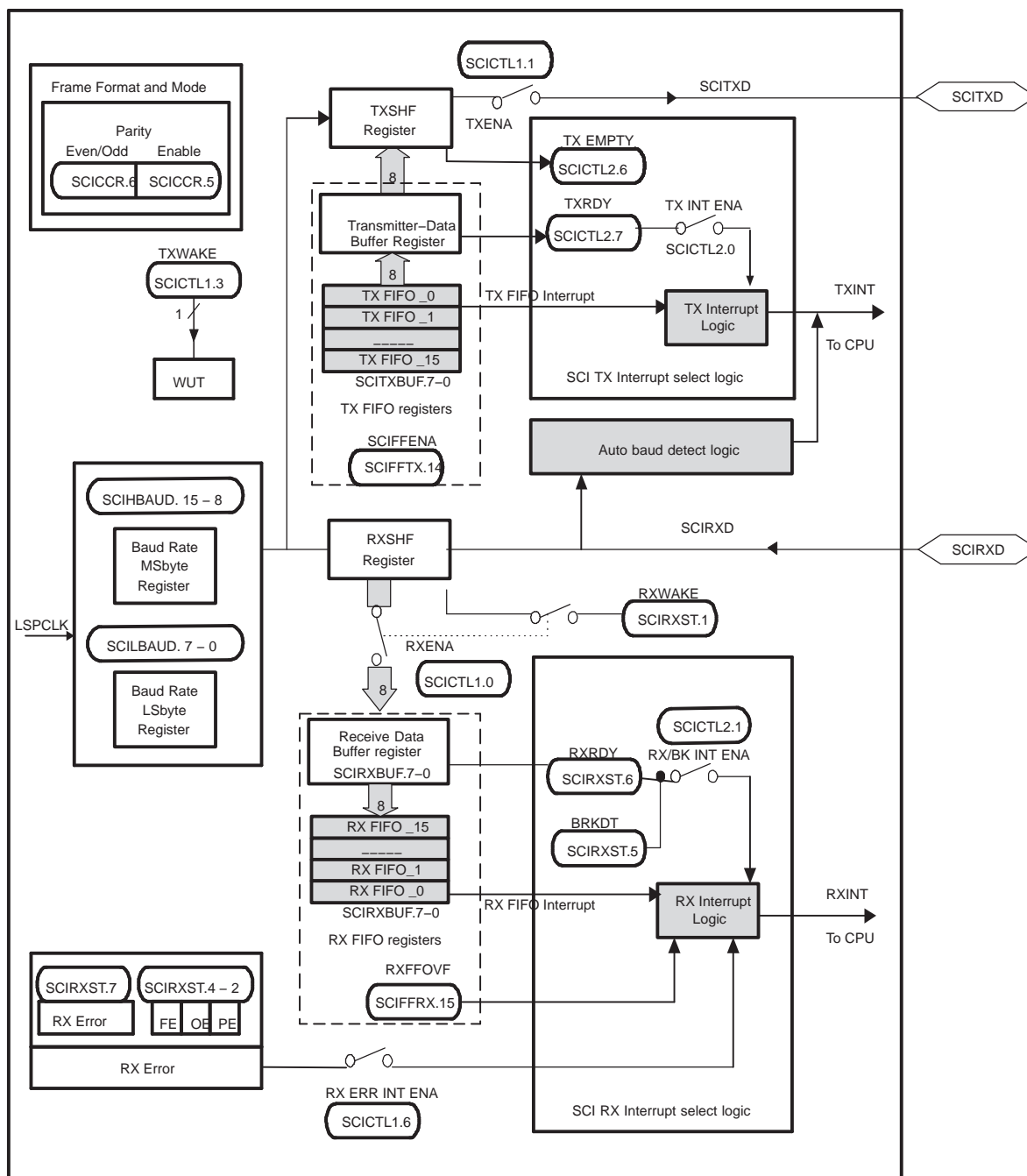
All registers in this module are 8-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7–0), and the upper byte (15–8) is read as zeros. Writing to the upper byte has no effect.

Enhanced features:

- Auto-baud-detect hardware logic
- 16-level transmit/receive FIFO

Figure 18-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Section 18.14 of this chapter.

Figure 18-2. Serial Communications Interface (SCI) Module Block Diagram



18.2 Architecture

The major elements used in full-duplex operation are shown in Figure 18-2 and include:

- A transmitter (TX) and its major registers (upper half of Figure 18-2)

- 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 18-2](#))
 - 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
- Data-memory-mapped control and status registers

The SCI receiver and transmitter can operate either independently or simultaneously.

18.3 SCI Module Signal Summary

Table 18-1. SCI Module Signal Summary

Signal Name	Description
External signals	
SCIRXD	SCI Asynchronous Serial Port receive data
SCITXD	SCI Asynchronous Serial Port transmit data
Control	
Baud clock	LSPCLK Prescaled clock
Interrupt signals	
TXINT	Transmit interrupt
RXINT	Receive Interrupt

18.4 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.

18.5 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the idle-line multiprocessor mode (see [Section 18.8](#)) and the address-bit multiprocessor mode (see [Section 18.9](#)). 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 18.10](#)) 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

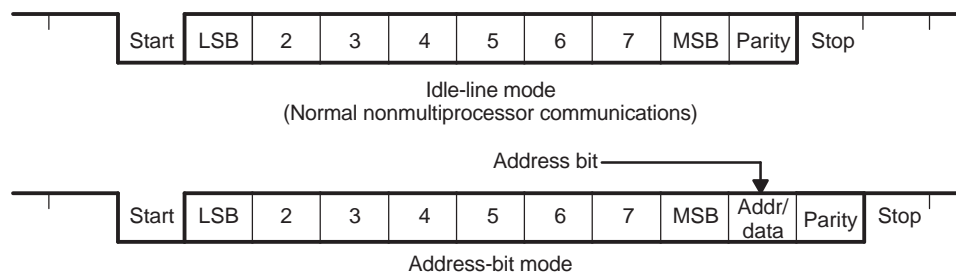
18.6 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (non-return-to-zero) format. The NRZ data format, shown in [Figure 18-3](#), 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 18-3](#).

Figure 18-3. 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 18-2](#).

Table 18-2. 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.

18.7 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.

18.7.1 Recognizing the Address Byte

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The idle-line mode ([Section 18.8](#)) 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 18.9](#)) 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.

18.7.2 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.

18.7.3 Receipt Sequence

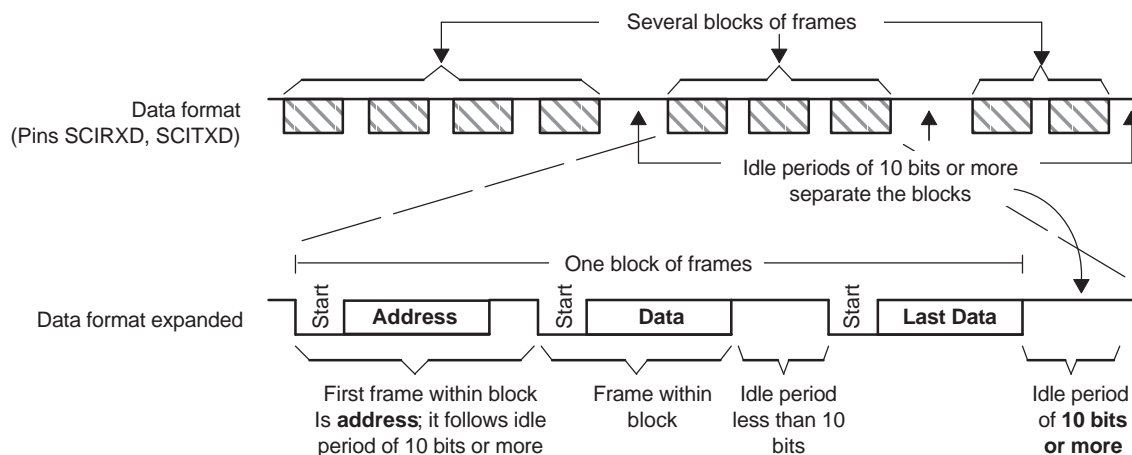
In both multiprocessor modes, the receive sequence is as follows:

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.

18.8 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 18-4](#) (ADDR/IDLE MODE bit is bit 3 of SCICCR).

Figure 18-4. Idle-Line Multiprocessor Communication Format



18.8.1 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.

18.8.2 Block Start Signal

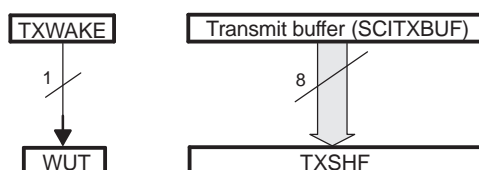
There are two ways to send a block-start signal:

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.)

18.8.3 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 18-5.

Figure 18-5. Double-Buffered WUT and TXSHF



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.

18.8.4 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.

18.9 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 18-6](#)).

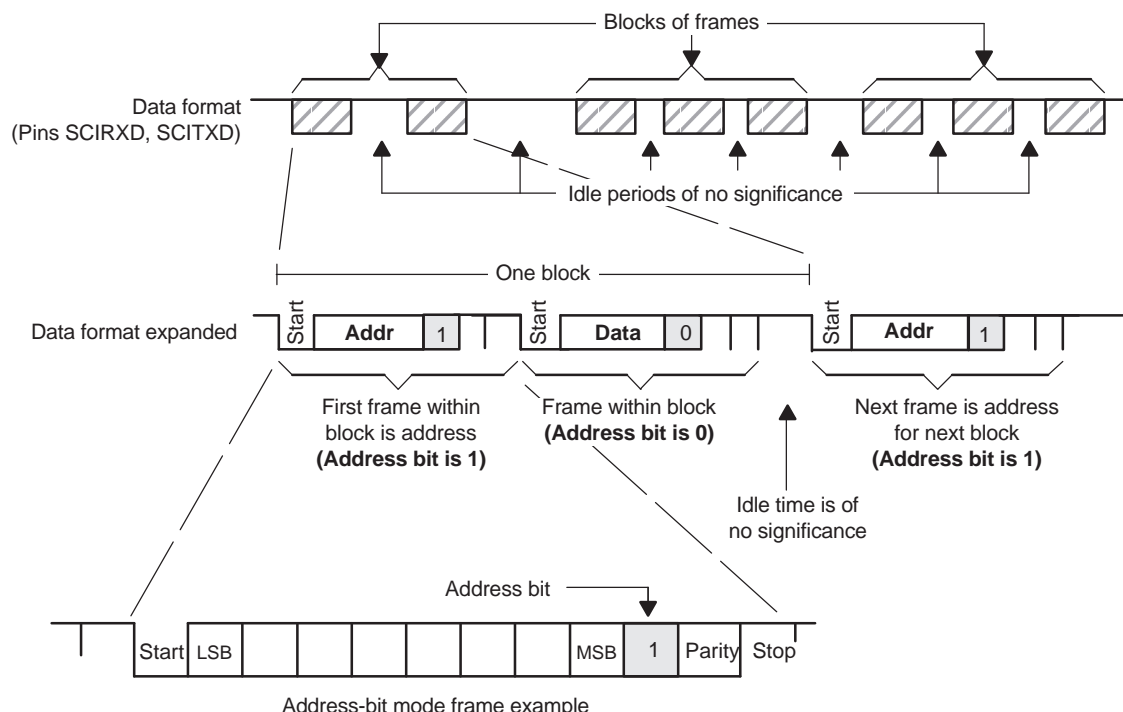
18.9.1 Sending an Address

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.
2. 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.)
3. 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 18-6. Address-Bit Multiprocessor Communication Format



18.10 SCI Communication Format

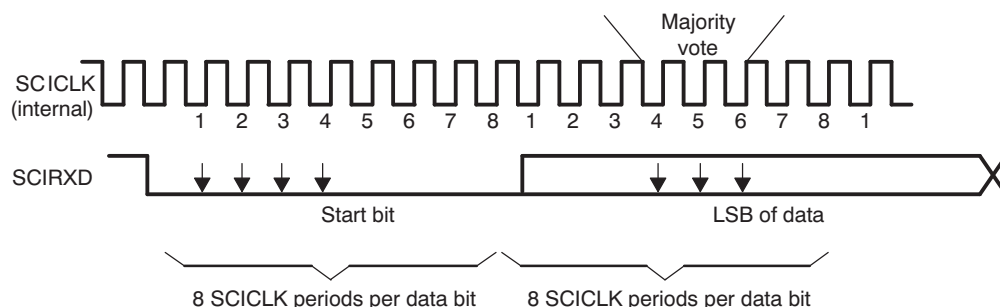
The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in [Figure 18-7](#)). There are eight SCICLK periods 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 18-7](#). 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 18-7](#) 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.

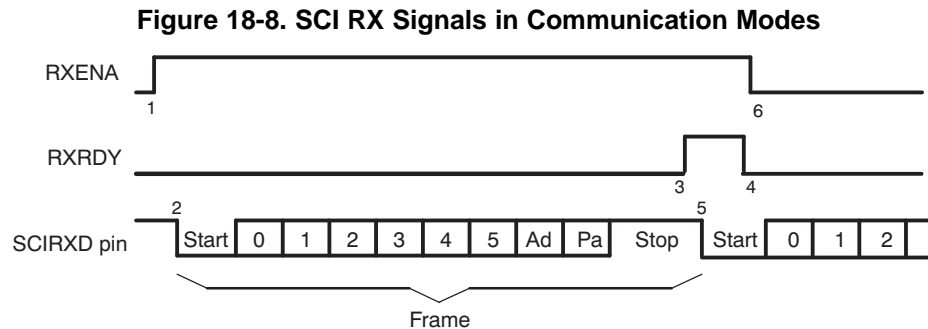
Figure 18-7. SCI Asynchronous Communications Format



18.10.1 Receiver Signals in Communication Modes

Figure 18-8 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



- (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.

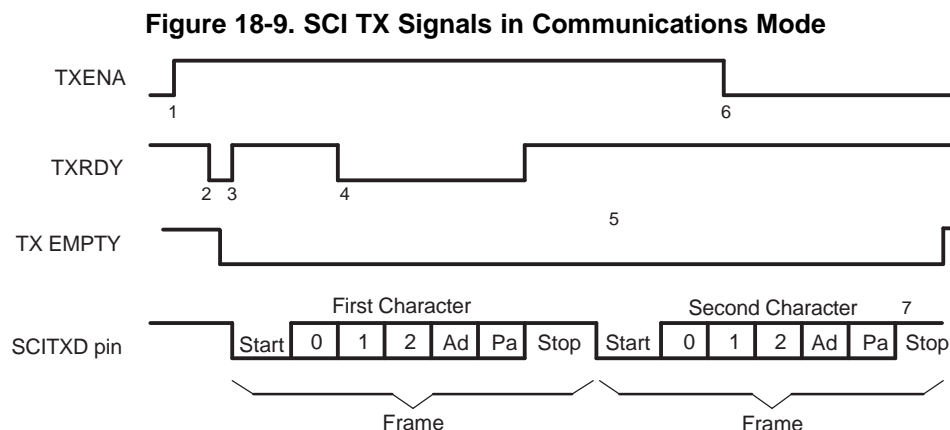
Notes:

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.

18.10.2 Transmitter Signals in Communication Modes

Figure 18-9 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.

18.11 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).

18.12 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 18-3](#) shows the baud-select values for common SCI bit rates.

Table 18-3. Asynchronous Baud Register Values for Common SCI Bit Rates

LSPCLK Clock Frequency, 200 MHz			
Ideal Baud	BRR	Actual Baud	% Error
2400	1952 (7A0h)	2400	0
4800	976 (3D0h)	4798	-0.04

Table 18-3. Asynchronous Baud Register Values for Common SCI Bit Rates (continued)

Ideal Baud	LSPCLK Clock Frequency, 200 MHz		
	BRR	Actual Baud	% Error
9600	487 (1E7h)	9606	0.06
19200	243 (F3h)	19211	0.06
38400	121 (79h)	38422	0.06

18.13 SCI Enhanced Features

The 28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

18.13.1 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 F24x 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) FTXDLY7–FTXDLY0 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 18-10 and Table 18-4 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.

Figure 18-10. SCI FIFO Interrupt Flags and Enable Logic

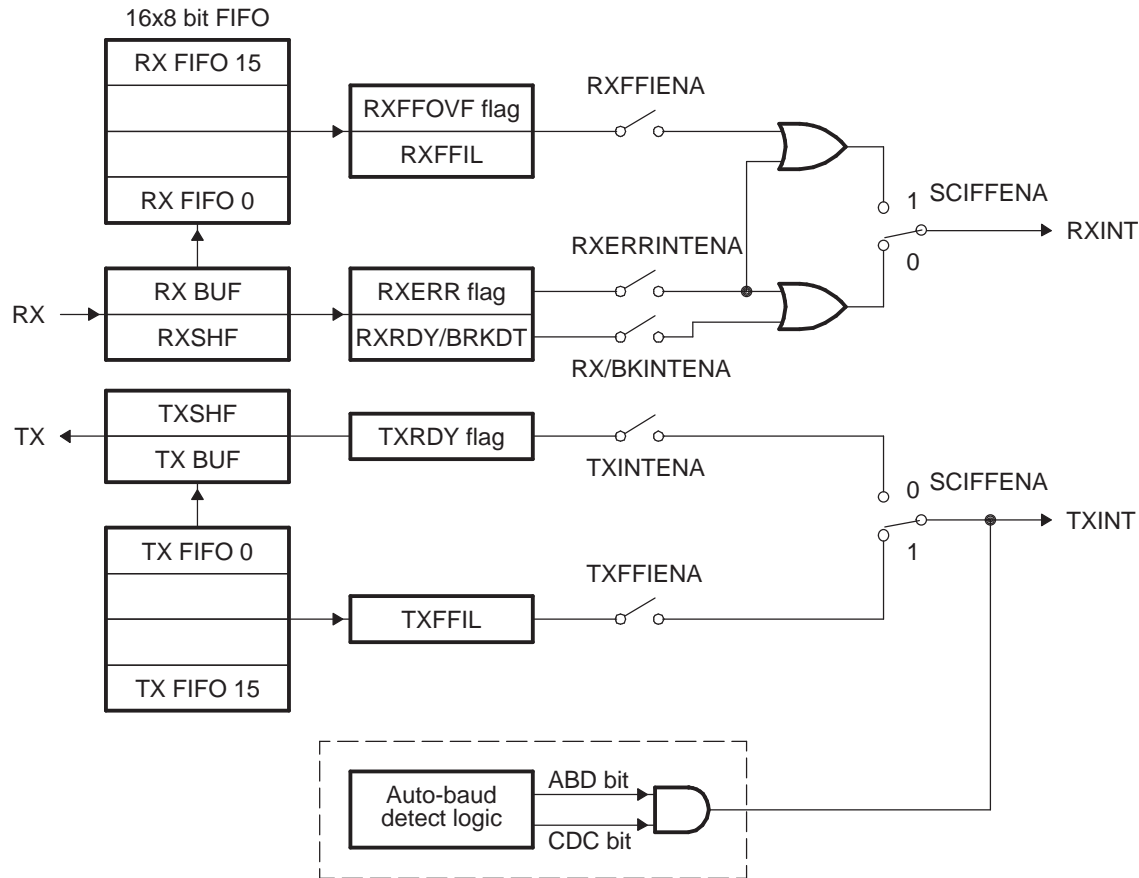


Table 18-4. SCI Interrupt Flags

FIFO Options ⁽¹⁾	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt Line
SCI without FIFO	Receive error	RXERR ⁽²⁾	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

⁽¹⁾ FIFO mode TXSHF is directly loaded after delay value, TXBUF is not used.

⁽²⁾ RXERR can be set by BRKDT, FE, OE, PE flags. In FIFO mode, BRKDT interrupt is only through RXERR flag

18.13.2 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.

18.13.3 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.
-

18.14 Registers

18.14.1 SCI Base Addresses

Table 18-5. 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
ScicRegs	SCI_REGS	0x0000_7220	0x0000_722F
ScidRegs	SCI_REGS	0x0000_7230	0x0000_723F

18.14.2 SCI_REGS Registers

Table 18-6 lists the memory-mapped registers for the SCI_REGS. All register offset addresses not listed in Table 18-6 should be considered as reserved locations and the register contents should not be modified.

Table 18-6. SCI_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	SCICCR	Communications control register		Go
1h	SCICTL1	Control register 1		Go
2h	SCIHBAUD	Baud rate (high) register		Go
3h	SCILBAUD	Baud rate (low) register		Go
4h	SCICTL2	Control register 2		Go
5h	SCIRXST	Receive status register		Go
6h	SCIRXEMU	Receive emulation buffer register		Go
7h	SCIRXBUF	Receive data buffer		Go
9h	SCITXBUF	Transmit data buffer		Go
Ah	SCIFFTX	FIFO transmit register		Go
Bh	SCIFFRX	FIFO receive register		Go
Ch	SCIFFCT	FIFO control register		Go
Fh	SCIPRI	FIFO Priority control		Go

18.14.2.1 SCICCR Register (Offset = 0h) [reset = 0h]

SCICCR is shown in [Figure 18-11](#) and described in [Table 18-7](#).

Communications control register

Figure 18-11. 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	SCICHAR		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-7. 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. 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). 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. 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. 0h (R/W) = Loop Back test mode disabled 1h (R/W) = Loop Back test mode enabled
3	ADDRIDLE_MODE	R/W	0h	SCI multiprocessor mode control bit. 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. 0h (R/W) = Idle-line mode protocol selected 1h (R/W) = Address-bit mode protocol selected

Table 18-7. SCICCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2-0	SCICCHAR	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>0h (R/W) = SCICCHAR_LEGNTNTH_1 1h (R/W) = SCICCHAR_LEGNTNTH_2 2h (R/W) = SCICCHAR_LEGNTNTH_3 3h (R/W) = SCICCHAR_LEGNTNTH_4 4h (R/W) = SCICCHAR_LEGNTNTH_5 5h (R/W) = SCICCHAR_LEGNTNTH_6 6h (R/W) = SCICCHAR_LEGNTNTH_7 7h (R/W) = SCICCHAR_LEGNTNTH_8</p>

18.14.2.2 SCICTL1 Register (Offset = 1h) [reset = 0h]

SCICTL1 is shown in [Figure 18-12](#) and described in [Table 18-8](#).

Control register 1

Figure 18-12. SCICTL1 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RXERRINTEN A	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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-8. SCICTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	RXERRINTENA	R/W	0h	SCI receive error interrupt enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring. 0h (R/W) = Receive error interrupt disabled 1h (R/W) = Receive error interrupt enabled
5	SWRESET	R/W	0h	SCI software reset (active low). 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. 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). 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. The affected flags are as follows: Value After SW SCI Flag Register Bit RESET 1 TXRDY SCICTL2, bit 7 1 TX EMPTY SCICTL2, bit 6 0 RXWAKE SCIRXST, bit 1 0 PE SCIRXST, bit 2 0 OE SCIRXST, bit 3 0 FE SCIRXST, bit 4 0 BRKDT SCIRXST, bit 5 0 RXRDY SCIRXST, bit 6 0 RX ERROR SCIRXST, bit 7 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. 1h (R/W) = After a system reset, re-enable the SCI by writing a 1 to this bit.
4	RESERVED	R	0h	Reserved

Table 18-8. SCICTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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>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)</p> <p>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>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.</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>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>

18.14.2.3 SCIHBAUD Register (Offset = 2h) [reset = 0h]

SCIHBAUD is shown in [Figure 18-13](#) and described in [Table 18-9](#).

Baud rate (high) register

Figure 18-13. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-9. 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>The SCI baud rate is calculated using the following equation: SCI Asynchronous Baud = $LSPCLK / ((BRR + 1) * 8)$ Alternatively, $BRR = LSPCLK / (SCI \text{ Asynchronous Baud} * 8) - 1$ Note that the above formulas are applicable only when $0 < BRR < 65536$. If $BRR = 0$, then SCI Asynchronous Baud = $LSPCLK / 16$ Where: BRR = the 16-bit value (in decimal) in the baud-select registers</p>

18.14.2.4 SCILBAUD Register (Offset = 3h) [reset = 0h]

SCILBAUD is shown in [Figure 18-14](#) and described in [Table 18-10](#).

Baud rate (low) register

Figure 18-14. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-10. SCILBAUD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	BAUD	R/W	0h	See SCILBAUD Detailed Description

18.14.2.5 SCICTL2 Register (Offset = 4h) [reset = 0h]

SCICTL2 is shown in [Figure 18-15](#) and described in [Table 18-11](#).

Control register 2

Figure 18-15. 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-0h	R-0h	R-0h				R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-11. SCICTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	TXRDY	R	0h	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. 0h (R/W) = SCITXBUF is full 1h (R/W) = SCITXBUF is ready to receive the next character
6	TXEMPTY	R	0h	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. 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. 0h (R/W) = Disable RXRDY/BRKDT interrupt 1h (R/W) = Enable RXRDY/BRKDT interrupt
0	TXINTENA	R/W	0h	SCITXBUF-register interrupt enable. This bit controls the interrupt request caused by setting the TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set (being set indicates that register SCITXBUF is ready to receive another character). 0h (R/W) = Disable TXRDY interrupt 1h (R/W) = Enable TXRDY interrupt

18.14.2.6 SCIRXST Register (Offset = 5h) [reset = 0h]

SCIRXST is shown in [Figure 18-16](#) and described in [Table 18-12](#).

Recieve status register

Figure 18-16. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-12. 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>0h (R/W) = No error flags set</p> <p>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>0h (R/W) = No new character in SCIRXBUF</p> <p>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>0h (R/W) = No break condition</p> <p>1h (R/W) = Break condition occurred</p>
4	FE	R	0h	<p>SCI framing-error flag.</p> <p>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.</p> <p>0h (R/W) = No framing error detected</p> <p>1h (R/W) = Framing error detected</p>

Table 18-12. SCIRXST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	OE	R	0h	<p>SCI overrun-error flag.</p> <p>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.</p> <p>0h (R/W) = No overrun error detected 1h (R/W) = Overrun error detected</p>
2	PE	R	0h	<p>SCI parity-error flag.</p> <p>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.</p> <p>0h (R/W) = No parity error or parity is disabled 1h (R/W) = Parity error is detected</p>
1	RXWAKE	R	0h	<p>Receiver wake-up-detect flag</p> <p>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:</p> <ul style="list-style-type: none"> - The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode) - The reading of SCIRXBUF - An active SW RESET - A system reset
0	RESERVED	R	0h	Reserved

18.14.2.7 SCIRXEMU Register (Offset = 6h) [reset = 0h]

SCIRXEMU is shown in [Figure 18-17](#) and described in [Table 18-13](#).

Receive emulation buffer register

Figure 18-17. SCIRXEMU Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
ERXDT							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-13. SCIRXEMU Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	ERXDT	R	0h	Receive emulation buffer data

18.14.2.8 SCIRXBUF Register (Offset = 7h) [reset = 0h]

SCIRXBUF is shown in [Figure 18-18](#) and described in [Table 18-14](#).

Recieve data buffer

Figure 18-18. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-14. 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) 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.
14	SCIFFPE	R	0h	SCIFFPE. SCI FIFO parity error flag bit (applicable only if the FIFO is enabled) 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.
13-8	RESERVED	R	0h	Reserved
7-0	SAR	R	0h	Receive Character bits

18.14.2.9 SCITXBUF Register (Offset = 9h) [reset = 0h]

SCITXBUF is shown in [Figure 18-19](#) and described in [Table 18-15](#).

Transmit data buffer

Figure 18-19. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-15. SCITXBUF Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	TXDT	R/W	0h	Transmit data buffer

18.14.2.10 SCIFFTX Register (Offset = Ah) [reset = 0h]

SCIFFTX is shown in [Figure 18-20](#) and described in [Table 18-16](#).

FIFO transmit register

Figure 18-20. SCIFFTX Register

15	14	13	12	11	10	9	8
SCIRST	SCIFFENA	TXFIFOXRESET	TXFFST				
R/W-0h	R/W-0h	R/W-0h	R-0h				
7	6	5	4	3	2	1	0
TXFFINT	TXFFINTCLR	TXFFIENA	TXFFIL				
R-0h	R-0h	R/W-0h	R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-16. SCIFFTX Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SCIRST	R/W	0h	SCI Reset
14	SCIFFENA	R/W	0h	SCI FIFO enable 0h (R/W) = SCI FIFO enhancements are disabled 1h (R/W) = SCI FIFO enhancements are enabled
13	TXFIFOXRESET	R/W	0h	Transmit FIFO reset 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 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 0h (R/W) = TXFIFO interrupt has not occurred, read-only bit 1h (R/W) = TXFIFO interrupt has occurred, read-only bit
6	TXFFINTCLR	R	0h	Transmit FIFO clear 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 0h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) is disabled 1h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) is enabled.

Table 18-16. SCIFFTX Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4-0	TXFFIL	R/W	0h	<p>TXFFIL4-0 Transmit FIFO interrupt level bits.</p> <p>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 RX 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.</p>

18.14.2.11 SCIFFRX Register (Offset = Bh) [reset = 1Fh]

SCIFFRX is shown in [Figure 18-21](#) and described in [Table 18-17](#).

FIFO receive register

Figure 18-21. SCIFFRX Register

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVRCLR	RXFIFORESET	RXFFST				
R/W-0h	R/W-0h	R/W-0h	R-0h				
7	6	5	4	3	2	1	0
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL				
R-0h	W-0h	R/W-0h	R/W-1Fh				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-17. SCIFFRX Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RXFFOVF	R/W	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. 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/W	0h	RXFFOVF clear 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	0h	Receive FIFO reset 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 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
7	RXFFINT	R	0h	Receive FIFO interrupt 0h (R/W) = RXFIFO interrupt has not occurred, read-only bit 1h (R/W) = RXFIFO interrupt has occurred, read-only bit

Table 18-17. SCIFFRX Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	RXFFINTCLR	W	0h	Receive FIFO interrupt clear 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 0h (R/W) = RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled 1h (R/W) = RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled
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.

18.14.2.12 SCIFFCT Register (Offset = Ch) [reset = 0h]

SCIFFCT is shown in [Figure 18-22](#) and described in [Table 18-18](#).

FIFO control register

Figure 18-22. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-18. SCIFFCT Register Field Descriptions

Bit	Field	Type	Reset	Description
15	ABD	R	0h	Auto-baud detect (ABD) bit 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 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 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.

18.14.2.13 SCIPRI Register (Offset = Fh) [reset = 0h]

SCIPRI is shown in [Figure 18-23](#) and described in [Table 18-19](#).

FIFO Priority control

Figure 18-23. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 18-19. 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	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. 0h (R/W) = Immediate stop on suspend 1h (R/W) = Complete current receive/transmit sequence before stopping 2h (R/W) = Free run 3h (R/W) = Free run
2-0	RESERVED	R	0h	Reserved

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 Philips 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.

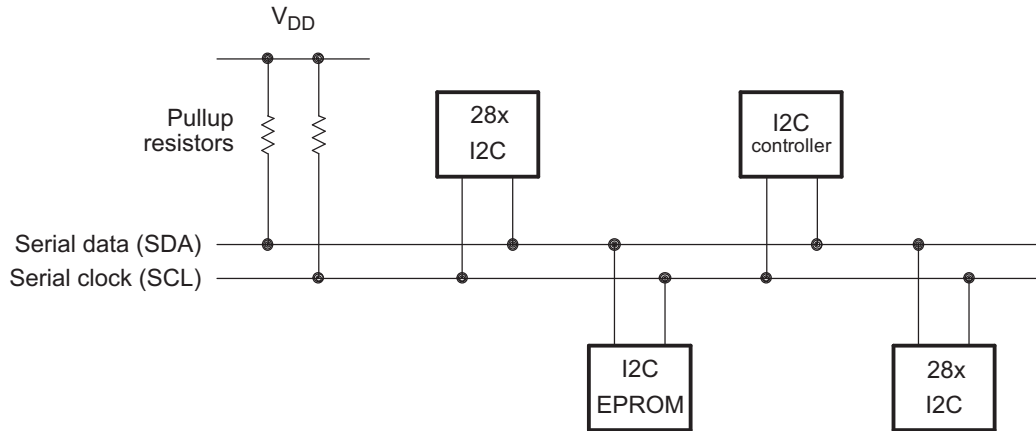
NOTE: A unit of data transmitted or received by the I2C module can have fewer than 8 bits; however, for convenience, a unit of data is called a data byte throughout this document. The number of bits in a data byte is selectable via the BC bits of the mode register, I2CMR.

Topic	Page
19.1 Introduction to the I2C Module	1952
19.2 Configuring Device Pins	1956
19.3 I2C Module Operational Details.....	1956
19.4 Interrupt Requests Generated by the I2C Module.....	1963
19.5 Resetting or Disabling the I2C Module.....	1965
19.6 Registers	1966

19.1 Introduction to the I2C Module

The I2C module supports any slave or master I2C-compatible device. [Figure 19-1](#) shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.

Figure 19-1. Multiple I2C Modules Connected



19.1.1 Features

The I2C module has the following features:

- Compliance with the Philips 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 of from 10 kbps up to 400 kbps (Philips Fast-mode rate)
- One 16-byte receive FIFO and one 16-byte transmit FIFO
- One interrupt that can always be used by the CPU. This interrupt can be generated as a result of one of the following conditions: transmit-data ready, receive-data ready, register-access ready, no-acknowledgment received, arbitration lost, stop condition detected, addressed as slave.
- An additional interrupt that can be used by the CPU when in FIFO mode
- Module enable/disable capability
- Free data format mode

19.1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode

19.1.3 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 the *Registers* section. 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:

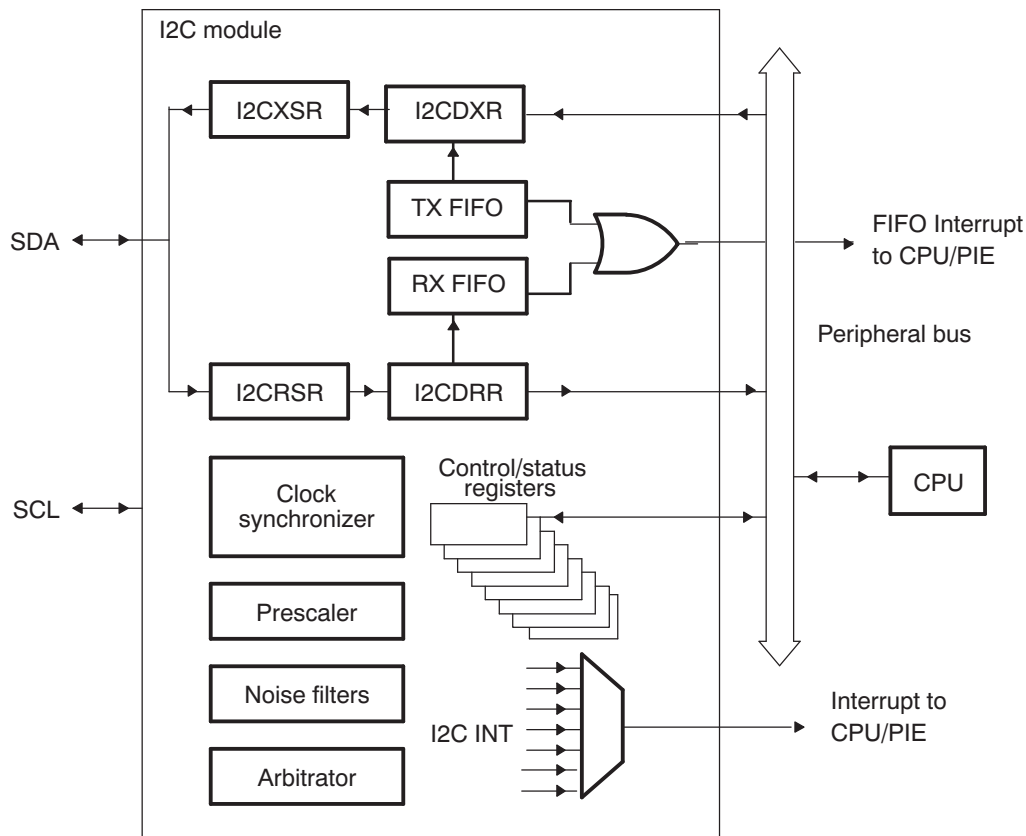
- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See the *Register* section for more information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See *Registers* for RM bit information.

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 19-2 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.

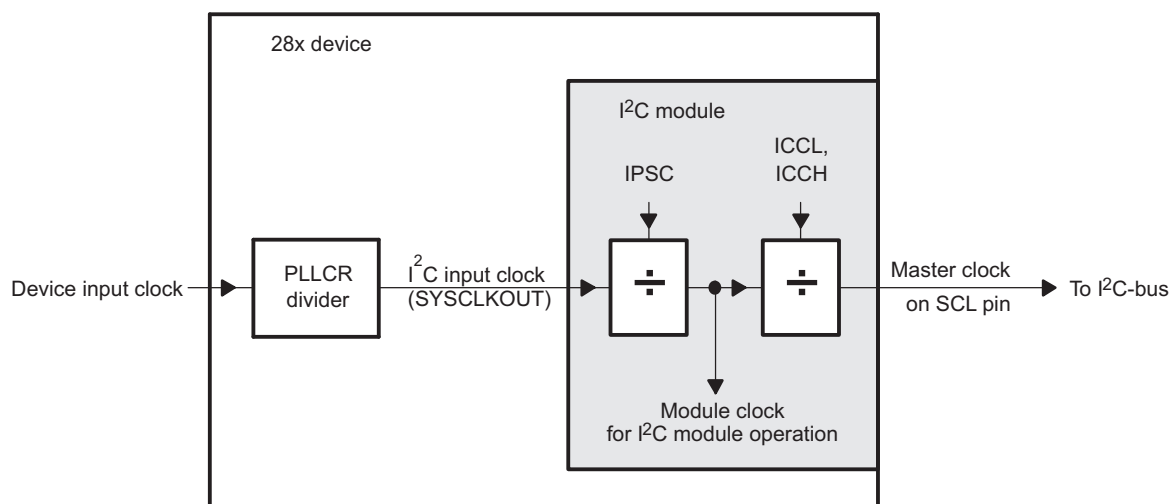
Figure 19-2. I2C Module Conceptual Block Diagram



19.1.4 Clock Generation

As shown in Figure 19-3, the device clock generator receives a signal from an external clock source and produces an I2C input clock with a programmed frequency. The I2C input clock is equivalent to the CPU clock and is then divided twice more inside the I2C module to produce the module clock and the master clock.

Figure 19-3. Clocking Diagram for the I2C Module



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the I2C input clock to produce the module clock. To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{module clock frequency} = \frac{\text{I2C input clock frequency}}{(\text{IPSC} + 1)}$$

NOTE: To meet all of the I2C protocol timing specifications, the module clock must be configured 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.

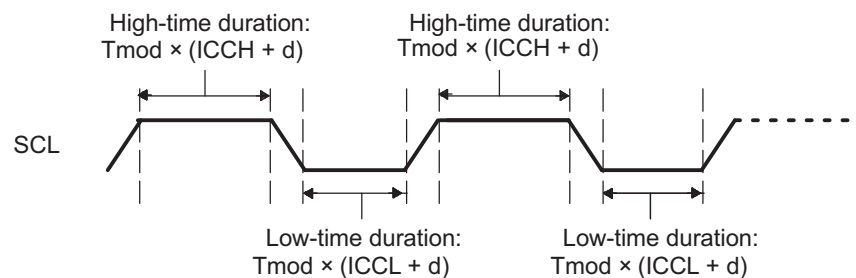
The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C-bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 19-3, a second clock divider in the I2C module divides down the module clock to produce the master 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 19.1.5 for the master clock frequency equation.

19.1.5 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in Section 19.1.4, when the I2C module is a master, the module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 19-4, 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 19-4. The Roles of the Clock Divide-Down Values (ICCL and ICCH)



19.1.5.1 Formula for the Master Clock Period

The period of the master clock (T_{mst}) is a multiple of the period of the module clock (T_{mod}):

$$T_{\text{mst}} = T_{\text{mod}} \times [(\text{ICCL} + d) + (\text{ICCH} + d)]$$

$$T_{\text{mst}} = \frac{(\text{IPSC} + 1) [(\text{ICCL} + d) + (\text{ICCH} + d)]}{\text{I2C input clock frequency}}$$

where d depends on the divide-down value IPSC, as shown in Table 19-1. IPSC is described in the I2CPSC register.

Table 19-1. Dependency of Delay d on the Divide-Down Value IPSC

IPSC	d
0	7
1	6
Greater than 1	5

19.2 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.

19.3 I2C Module Operational Details

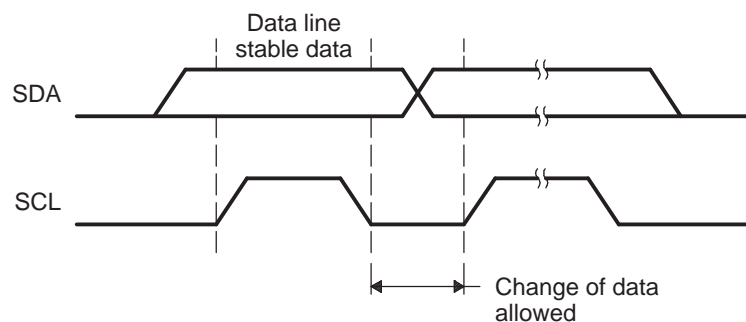
This section provides an overview of the I2C-bus protocol and how it is implemented.

19.3.1 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.

19.3.2 Data Validity

The data on SDA must be stable during the high period of the clock (see [Figure 19-5](#)). The high or low state of the data line, SDA, should change only when the clock signal on SCL is low.

Figure 19-5. Bit Transfer on the I2C-Bus


19.3.3 Operating Modes

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See [Table 19-2](#) 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 19-2. 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 (RSFULL = 1 in I2CSTR) after a byte has been received. See section Section 19.3.7 for more details.</p>
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 $R/\overline{W} = 1$. 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 (XSMT = 0 in I2CSTR) after a byte has been transmitted. See section Section 19.3.7 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 $R/\overline{W} = 1$. 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 (RSFULL = 1 in I2CSTR) after a byte has been received.</p>
Master-transmitter modes	<p>The IC 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 (XSMT = 0 in I2CSTR) after a byte has been transmitted.</p>

Table 19-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR

RM	STT	STP	Bus Activity ⁽¹⁾	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)

⁽¹⁾ S = START condition; A = Address; D = Data byte; P = STOP condition;

19.3.4 I2C Module START and STOP Conditions

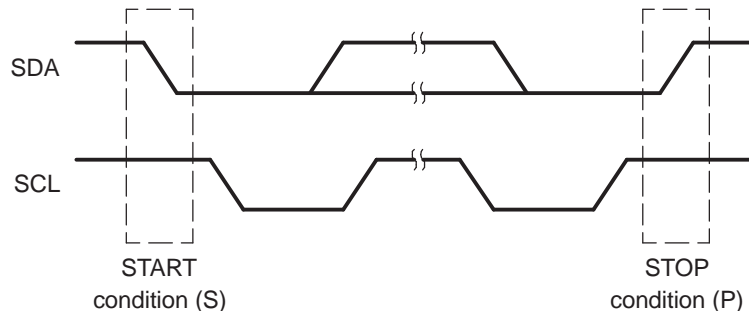
START and STOP conditions can be generated by the I2C module when the module is configured to be a master on the I2C-bus. As shown in [Figure 19-6](#):

- 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 19-6. 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*.

19.3.5 Serial Data Formats

Figure 19-7 shows an example of a data transfer on the I2C-bus. The I2C module supports 1 to 8-bit data values. In Figure 19-7, 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 19-7 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 19-8 through Figure 19-10 and described in the paragraphs that follow the figures.

NOTE: In Figure 19-7 through Figure 19-10, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

Figure 19-7. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)

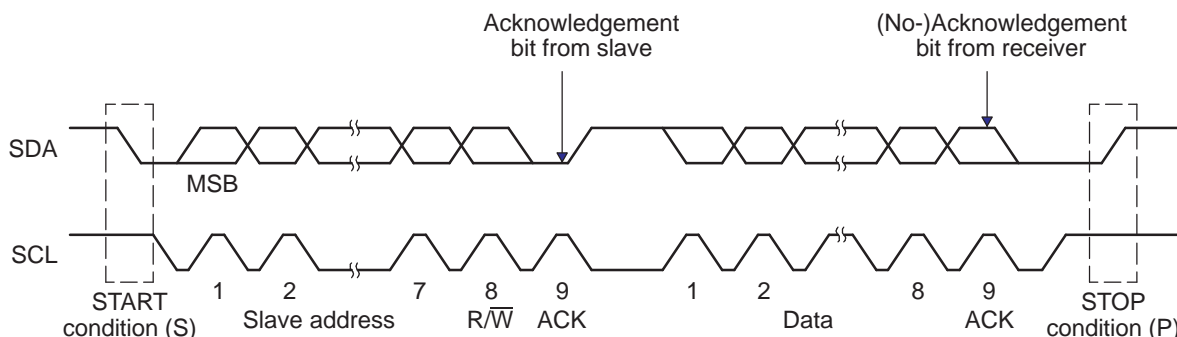


Figure 19-8. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)

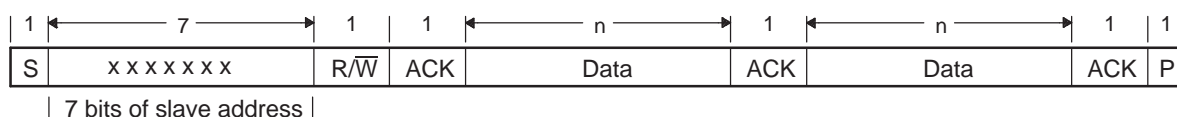


Figure 19-9. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)

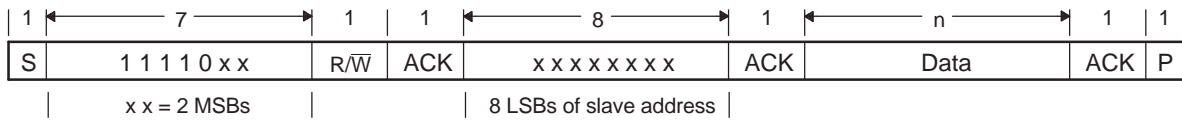
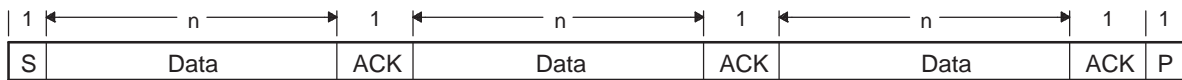


Figure 19-10. I2C Module Free Data Format (FDF = 1 in I2CMDR)



19.3.5.1 7-Bit Addressing Format

In the 7-bit addressing format (see [Figure 19-8](#)), 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 master writes (transmits) data to the addressed slave.
- R/W = 1: The master reads (receives) data from the slave.

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.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR, and make sure the free data format mode is off (FDF = 0 in I2CMDR).

19.3.5.2 10-Bit Addressing Format

The 10-bit addressing format (see [Figure 19-9](#)) 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 = 0 (write). 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 Philips Semiconductors I2C-bus specification.

To select the 10-bit addressing format, write 1 to the XA bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

19.3.5.3 Free Data Format

In this format (see [Figure 19-10](#)), 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.

To select the free data format, write 1 to the free data format (FDF) bit of I2CMDR. The free data format is not supported in the digital loopback mode (DLB = 1 in I2CMDR).

Table 19-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR

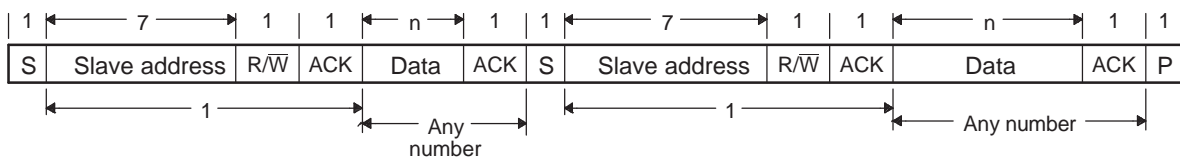
MST	FDF	I2C Module State	Function of TRX
0	0	In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C module responds as a receiver or a transmitter.
0	1	In slave mode and free data format mode	<p>The free data format mode requires that the I2C module remains the transmitter or the receiver throughout the transfer. TRX identifies the role of the I2C module:</p> <p>TRX = 1: The I2C module is a transmitter.</p> <p>TRX = 0: The I2C module is a receiver.</p>

Table 19-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR (continued)

MST	FDF	I2C Module State	Function of TRX
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.

19.3.5.4 Using a Repeated START Condition

At the end of each data byte, the master can drive another START condition. Using this capability, a master can communicate with multiple slave addresses without having to give up control of the bus by driving a STOP condition. The length of a data byte can be from 1 to 8 bits and is selected with the BC field of I2CMDR. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. [Figure 19-11](#) shows a repeated START condition in the 7-bit addressing format.

Figure 19-11. Repeated START Condition (in This Case, 7-Bit Addressing Format)


NOTE: In [Figure 19-11](#), n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

19.3.6 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 19-5](#) summarizes the various ways you can tell the I2C module to send a NACK bit.

Table 19-5. Ways to Generate a NACK Bit

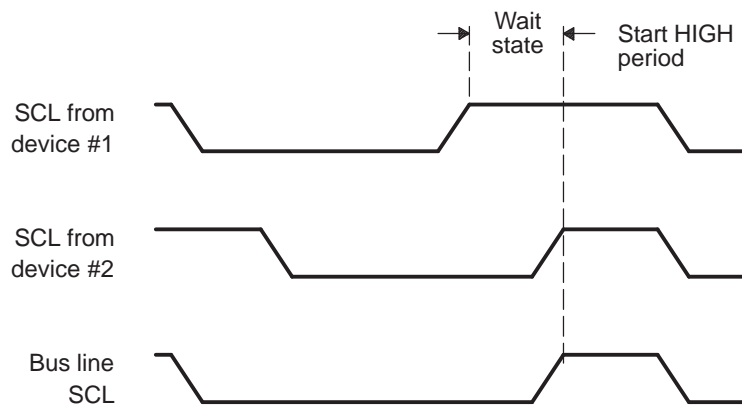
I2C Module Condition	NACK Bit Generation Options
Slave-receiver modes	<ul style="list-style-type: none"> Allow an overrun condition (RSFULL = 1 in I2CSTR) Reset the module (IRS = 0 in I2CMDR) Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Repeat mode (RM = 1 in I2CMDR)	<ul style="list-style-type: none"> Generate a STOP condition (STP = 1 in I2CMDR) Reset the module (IRS = 0 in I2CMDR) Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2CMDR)	<ul style="list-style-type: none"> If STP = 1 in I2CMDR, allow the internal data counter to count down to 0 and thus force a STOP condition If STP = 0, make STP = 1 to generate a STOP condition Reset the module (IRS = 0 in I2CMDR). = 1 to generate a STOP condition Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive

19.3.7 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 19-12 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 19-12. Synchronization of Two I2C Clock Generators During Arbitration



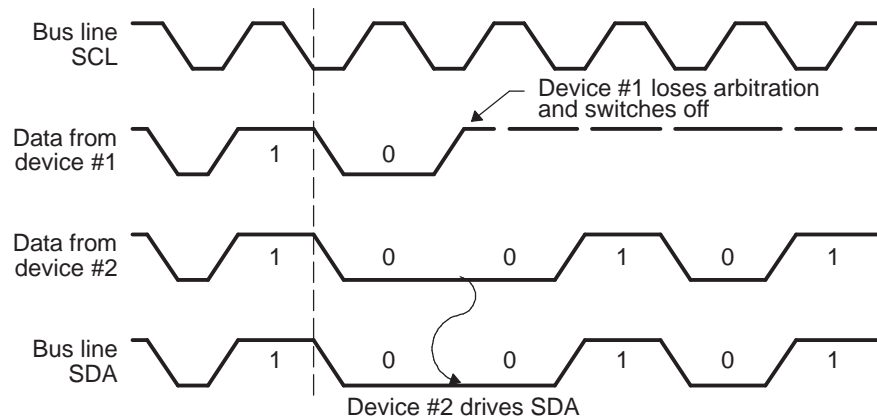
19.3.8 Arbitration

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 19-13 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 (AL) 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

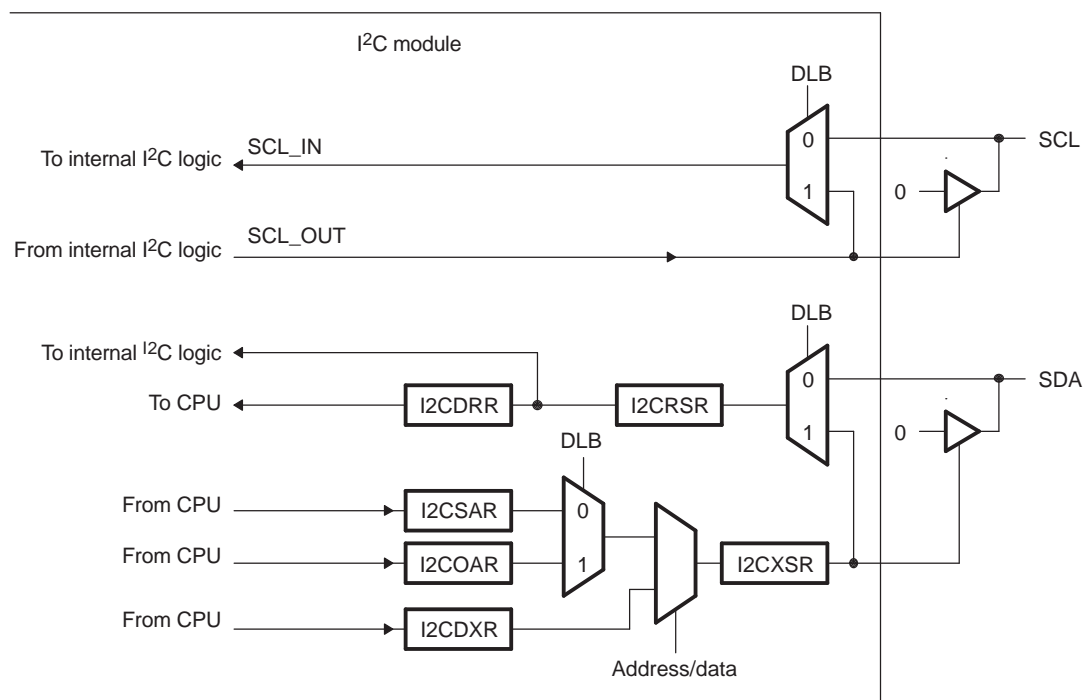
Figure 19-13. Arbitration Procedure Between Two Master-Transmitters


19.3.9 Digital Loopback Mode

The I2C module support a self-test mode called digital loopback, which is enabled by setting the DLB bit in the I2CMDR 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:

$$n = 8 * (\text{I2C input clock frequency}) / (\text{Module clock frequency})$$

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 19-14](#) shows the signal routing in digital loopback mode.

Figure 19-14. 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.

19.4 Interrupt Requests Generated by the I2C Module

The I2C module can generate seven types of basic interrupt requests, which are described in [Section 19.4.1](#). Two of these can tell the CPU when to write transmit data and when to read receive data. If you want the FIFOs to handle transmit and receive data, you can also use the FIFO interrupts described in [Section 19.4.2](#). The basic I2C interrupts are combined to form PIE Group 8, Interrupt 1 (I2CINT1A_ISR), and the FIFO interrupts are combined to form PIE Group 8, Interrupt 2 (I2CINT2A_ISR).

19.4.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in [Table 19-6](#). As shown in [Figure 19-15](#), 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.

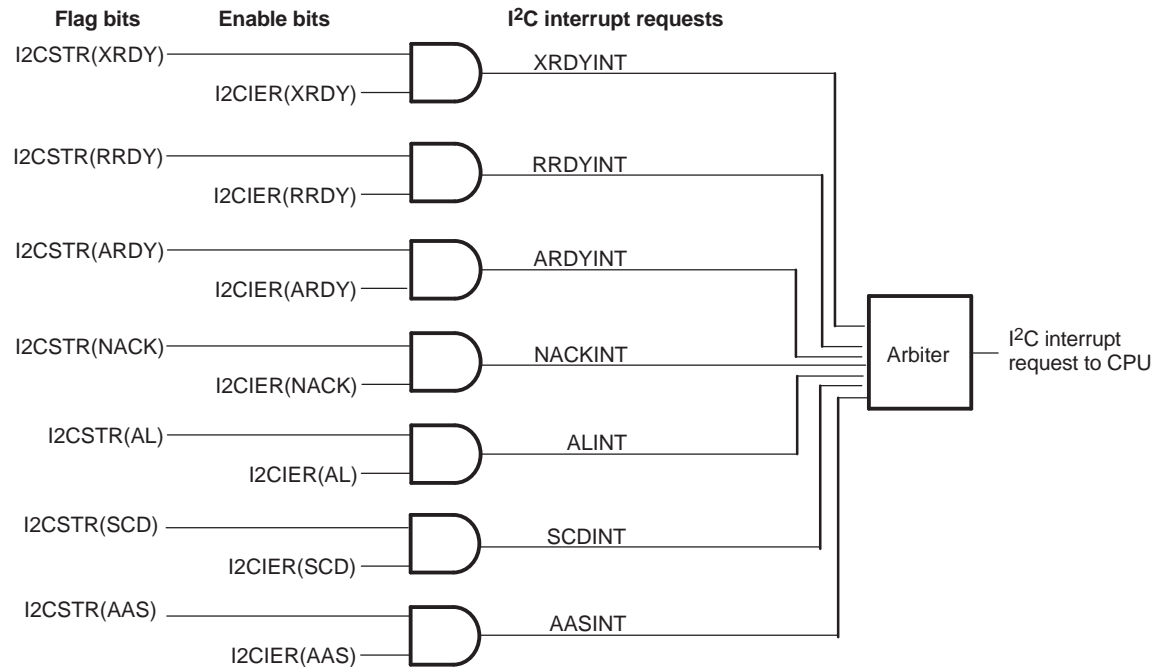
The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A_ISR). The I2CINT1A_ISR for the I2C interrupt can determine the interrupt source by reading 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 19-6. Descriptions of the Basic I2C Interrupt Requests

I2C Interrupt Request	Interrupt Source
XRDYINT	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). 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.
RRDYINT	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. 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.
ARDYINT	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. The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR. As an alternative to using ARDYINT, the CPU can poll the ARDY bit.
NACKINT	No-acknowledgment condition: The I2C module is configured as a master-transmitter and did not received acknowledgment from the slave-receiver. As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR.
ALINT	Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter. As an alternative to using ALINT, the CPU can poll the AL bit of I2CSTR.
SCDINT	Stop condition detected: A STOP condition was detected on the I2C bus. As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR.
AASINT	Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus. As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR.

Figure 19-15. Enable Paths of the I2C Interrupt Requests


19.4.2 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. 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.

19.5 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 $\overline{\text{XRS}}$ pin low. The entire device is reset and is held in the reset state until you drive the pin high. When the $\overline{\text{XRS}}$ 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.

19.6 Registers

19.6.1 I2C Base Addresses

Table 19-7. I2C Base Address Table

Device Registers	Register Names	Start Address	End Address
I2caRegs	I2C_REGS	0x0000_7300	0x0000_733F
I2cbRegs	I2C_REGS	0x0000_7340	0x0000_737F

19.6.2 I2C_REGS Registers

Table 19-8 lists the memory-mapped registers for the I2C_REGS. All register offset addresses not listed in Table 19-8 should be considered as reserved locations and the register contents should not be modified.

Table 19-8. I2C_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	I2COAR	I2C Own address		Go
1h	I2CIER	I2C Interrupt Enable		Go
2h	I2CSTR	I2C Status		Go
3h	I2CCLKL	I2C Clock low-time divider		Go
4h	I2CCLKH	I2C Clock high-time divider		Go
5h	I2CCNT	I2C Data count		Go
6h	I2CDRR	I2C Data receive		Go
7h	I2CSAR	I2C Slave address		Go
8h	I2CDXR	I2C Data Transmit		Go
9h	I2CMDR	I2C Mode		Go
Ah	I2CISRC	I2C Interrupt Source		Go
Bh	I2CEMDR	I2C Extended Mode		Go
Ch	I2CPSC	I2C Prescaler		Go
20h	I2CFFTX	I2C FIFO Transmit		Go
21h	I2CFFRX	I2C FIFO Receive		Go

19.6.2.1 I2COAR Register (Offset = 0h) [reset = 0h]

I2COAR is shown in [Figure 19-16](#) and described in [Table 19-9](#).

I2C Own address

Figure 19-16. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-9. I2COAR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OAR	R/W	0h	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. In 10-bit addressing mode (XA = 1 in I2CMDR): 000h-3FFh Bits 9-0 provide the 10-bit slave address of the I2C module.

19.6.2.2 I2CIER Register (Offset = 1h) [reset = 0h]

I2CIER is shown in [Figure 19-17](#) and described in [Table 19-10](#).

I2C Interrupt Enable

Figure 19-17. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-10. 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 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
5	SCD	R/W	0h	Stop condition detected interrupt enable 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. 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. 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
2	ARDY	R/W	0h	Register-access-ready interrupt enable 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
1	NACK	R/W	0h	No-acknowledgment interrupt enable 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
0	ARBL	R/W	0h	Arbitration-lost interrupt enable 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled

19.6.2.3 I2CSTR Register (Offset = 2h) [reset = 0h]

I2CSTR is shown in [Figure 19-18](#) and described in [Table 19-11](#).

I2C Status

Figure 19-18. I2CSTR Register

15	14	13	12	11	10	9	8
RESERVED	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-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
RESERVED		SCD	XRDY	RRDY	ARDY	NACK	ARBL
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-11. I2CSTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	SDIR	R/W	0h	Slave direction bit 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/W	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 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/W	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 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 19-11. I2CSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	RSFULL	R/W	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>0h (R/W) = No overrun detected. RSFULL is cleared by any one of the following events:</p> <ul style="list-style-type: none"> - I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit. - The I2C module is reset. <p>1h (R/W) = Overrun detected</p>
10	XSMT	R/W	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>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"> - Data is written to I2CDXR. - The I2C module is reset
9	AAS	R/W	0h	<p>Addressed-as-slave bit</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). The AAS bit is also set if the first byte has been received in the free data format (FDF = 1 in I2CMDR).</p>
8	AD0	R/W	0h	<p>Address 0 bits</p> <p>0h (R/W) = AD0 has been cleared by a START or STOP condition.</p> <p>1h (R/W) = An address of all zeros (general call) is detected.</p>
7-6	RESERVED	R/W	0h	Reserved
5	SCD	R/W	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>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"> - 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. - SCD is manually cleared. To clear this bit, write a 1 to it. - The I2C module is reset. <p>1h (R/W) = A STOP condition has been detected on the I2C bus.</p>

Table 19-11. I2CSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	XRDY	R/W	0h	<p>Transmit-data-ready interrupt flag bit.</p> <p>When not in FIFO mode, XRDY indicates that 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). The CPU can poll XRDY or use the XRDY interrupt request. When in FIFO mode, use TXFFINT instead.</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/W	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>0h (R/W) = I2CDRR not ready. RRDY is cleared by any one of the following events:</p> <ul style="list-style-type: none"> - I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit. - RRDY is manually cleared. To clear this bit, write a 1 to it. - The I2C module is reset. <p>1h (R/W) = I2CDRR ready: Data has been copied from I2CRSR to I2CDRR.</p>
2	ARDY	R/W	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>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"> - The I2C module starts using the current register contents. - ARDY is manually cleared. To clear this bit, write a 1 to it. - The I2C module is reset. <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>

Table 19-11. I2CSTR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	NACK	R/W	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>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"> - An acknowledge bit (ACK) has been sent by the receiver. - NACK is manually cleared. To clear this bit, write a 1 to it. - 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. - The I2C module is reset. <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/W	0h	<p>Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter).</p> <p>AL primarily indicates when the I2C module has lost an arbitration contest with another mastertransmitter. The CPU can poll AL or use the AL interrupt request</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"> - AL is manually cleared. To clear this bit, write a 1 to it. - The CPU reads the interrupt source register (I2CISRC) and the register contains the code for an <p>AL interrupt. Emulator reads of the I2CISRC do not affect this bit.</p> <ul style="list-style-type: none"> - The I2C module is reset. <p>1h (R/W) = Arbitration lost. AL is set by any one of the following events:</p> <ul style="list-style-type: none"> - The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. - The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1. <p>When AL becomes 1, the MST and STP bits of I2CMDR are cleared, and the I2C module becomes a slave-receiver.</p>

19.6.2.4 I2CCLKL Register (Offset = 3h) [reset = 0h]

I2CCLKL is shown in [Figure 19-19](#) and described in [Table 19-12](#).

I2C Clock low-time divider

Figure 19-19. I2CCLKL Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CCLKL															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-12. 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>

19.6.2.5 I2CCLKH Register (Offset = 4h) [reset = 0h]

I2CCLKH is shown in [Figure 19-20](#) and described in [Table 19-13](#).

I2C Clock high-time divider

Figure 19-20. I2CCLKH Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CCLKH															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-13. 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>

19.6.2.6 I2CCNT Register (Offset = 5h) [reset = 0h]

I2CCNT is shown in [Figure 19-21](#) and described in [Table 19-14](#).

I2C Data count

Figure 19-21. I2CCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CCNT															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-14. I2CCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	I2CCNT	R/W	0h	<p>Data count value. ICDC indicates the number of data bytes to transfer or receive.</p> <p>The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.</p> <p>The start value loaded to the internal data counter is 65536.</p> <p>The start value loaded to internal data counter is 1-65535.</p>

19.6.2.7 I2CDRR Register (Offset = 6h) [reset = 0h]

I2CDRR is shown in [Figure 19-22](#) and described in [Table 19-15](#).

I2C Data receive

Figure 19-22. I2CDRR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
DATA							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-15. I2CDRR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R	0h	Receive data

19.6.2.8 I2CSAR Register (Offset = 7h) [reset = 3FFh]

I2CSAR is shown in [Figure 19-23](#) and described in [Table 19-16](#).

I2C Slave address

Figure 19-23. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-16. 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.</p>

19.6.2.9 I2CDXR Register (Offset = 8h) [reset = 0h]

I2CDXR is shown in [Figure 19-24](#) and described in [Table 19-17](#).

I2C Data Transmit

Figure 19-24. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-17. I2CDXR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R/W	0h	Transmit data

19.6.2.10 I2CMDR Register (Offset = 9h) [reset = 0h]

I2CMDR is shown in [Figure 19-25](#) and described in [Table 19-18](#).

I2C Mode

Figure 19-25. I2CMDR 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-18. I2CMDR Register Field Descriptions

Bit	Field	Type	Reset	Description
15	NACKMOD	R/W	0h	<p>NACK mode bit.</p> <p>This bit is only applicable when the I2C module is acting as a receiver.</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>0h (R/W) = When I2C module is master:</p> <p>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:</p> <p>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>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 19-18. 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>0h (R/W) = STP is automatically cleared after the STOP condition has been generated</p> <p>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>0h (R/W) = Slave mode. The I2C module is a slave and receives the serial clock from the master.</p> <p>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>0h (R/W) = Receiver mode. The I2C module is a receiver and receives data on the SDA pin.</p> <p>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>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).</p> <p>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>0h (R/W) = Nonrepeat mode. The value in the data count register (I2CCNT) determines how many bytes are received/transmitted by the I2C module.</p> <p>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 19-18. I2CMDR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	DLB	R/W	0h	<p>Digital loopback mode bit.</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> $n = ((I2C \text{ input clock frequency} / \text{module clock frequency}) \div 8)$ <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>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>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"> 1. A START condition 2. A START byte (0000 0001b) 3. A dummy acknowledge clock pulse 4. A repeated START condition <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>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 19-18. 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>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>

19.6.2.11 I2CISRC Register (Offset = Ah) [reset = 0h]

I2CISRC is shown in [Figure 19-26](#) and described in [Table 19-19](#).

I2C Interrupt Source

Figure 19-26. I2CISRC Register

15	14	13	12	11	10	9	8
RESERVED				RESERVED			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED					INTCODE		
R-0h					R-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-19. I2CISRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R/W	0h	Reserved
7-3	RESERVED	R	0h	Reserved
2-0	INTCODE	R	0h	<p>Interrupt code bits.</p> <p>The binary code in INTCODE indicates the event that generated an I2C interrupt.</p> <p>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.</p> <p>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.</p> <p>Emulator reads will not affect the state of this field or of the status bits in the I2CSTR register.</p> <p>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</p>

19.6.2.12 I2CEMDR Register (Offset = Bh) [reset = 0h]

I2CEMDR is shown in [Figure 19-27](#) and described in [Table 19-20](#).

I2C Extended Mode

Figure 19-27. I2CEMDR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							BC
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-20. I2CEMDR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	BC	R/W	0h	Backwards compatibility mode. This bit affects the timing of the transmit status bits (XRDY and XSMT) in the I2CSTR register when in slave transmitter mode. 0h (R/W) = See Figure 9-17 for details. 1h (R/W) = See Figure 9-17 for details.

19.6.2.13 I2CPSC Register (Offset = Ch) [reset = 0h]

I2CPSC is shown in [Figure 19-28](#) and described in [Table 19-21](#).

I2C Prescaler

Figure 19-28. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-21. 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>

19.6.2.14 I2CFFTX Register (Offset = 20h) [reset = 0h]

I2CFFTX is shown in [Figure 19-29](#) and described in [Table 19-22](#).

I2C FIFO Transmit

Figure 19-29. 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/W-0h	R/W-0h	R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-22. 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. 0h (R/W) = Disable the I2C FIFO mode. 1h (R/W) = Enable the I2C FIFO mode.
13	TXFFRST	R/W	0h	Transmit FIFO Reset 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.
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. 0h (R/W) = Transmit FIFO interrupt condition has not occurred. 1h (R/W) = Transmit FIFO interrupt condition has occurred.
6	TXFFINTCLR	R/W	0h	Transmit FIFO Interrupt Flag Clear 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 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.
4-0	TXFFIL	R/W	0h	Transmit FIFO interrupt level. 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 these devices has a 16-level transmit FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.

19.6.2.15 I2CFFRX Register (Offset = 21h) [reset = 0h]

I2CFFRX is shown in [Figure 19-30](#) and described in [Table 19-23](#).

I2C FIFO Receive

Figure 19-30. 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/W-0h	R/W-0h	R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 19-23. 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 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.
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 0h (R/W) = Receive FIFO interrupt condition has not occurred. 1h (R/W) = Receive FIFO interrupt condition has occurred.
6	RXFFINTCLR	R/W	0h	Receive FIFO interrupt flag clear bit. 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. 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.
4-0	RXFFIL	R/W	0h	Receive FIFO interrupt level. 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. 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 these devices has a 16-level receive FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.

Multichannel Buffered Serial Port (McBSP)

This document describes the multichannel buffered serial port (McBSP) of this device.

Topic	Page
20.1 Overview.....	1990
20.2 Configuring Device Pins	1992
20.3 McBSP Operation.....	1992
20.4 McBSP Sample Rate Generator	2002
20.5 McBSP Exception/Error Conditions.....	2009
20.6 Multichannel Selection Modes	2017
20.7 SPI Operation Using the Clock Stop Mode	2024
20.8 Receiver Configuration	2031
20.9 Transmitter Configuration	2050
20.10 Emulation and Reset Considerations	2066
20.11 Data Packing Examples	2069
20.12 Interrupt Generation.....	2071
20.13 McBSP Modes	2073
20.14 McBSP Registers	2075

20.1 Overview

This device provides up to two high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system. The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in [Figure 20-1](#).

Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame synchronization), and FSR (receive frame synchronization).

The CPU and the DMA controller communicate with the McBSP through 16-bit-wide registers accessible via the internal peripheral bus. The CPU or the DMA controller writes the data to be transmitted to the data transmit registers (DXR1, DXR2). Data written to the DXRs is shifted out to DX via the transmit shift registers (XSR1, XSR2). Similarly, receive data on the DR pin is shifted into the receive shift registers (RSR1, RSR2) and copied into the receive buffer registers (RBR1, RBR2). The contents of the RBRs is then copied to the DRRs, which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted) if the serial word length is 8 bits, 12 bits, or 16 bits. For larger word lengths, these registers are needed to hold the most significant bits.

The frame and clock loop-back is implemented at chip level to enable CLKX and FSX to drive CLKR and FSR. If the loop-back is enabled, the CLKR and FSR get their signals from the CLKX and FSX pads; instead of the CLKR and FSR pins.

20.1.1 Features of the McBSPs

The McBSPs feature:

- Full-duplex communication
- Double-buffered transmission and triple-buffered reception, allowing a continuous data stream
- Independent clocking and framing for reception and transmission
- The capability to send interrupts to the CPU and to send DMA events to the DMA controller
- 128 channels for transmission and reception
- Multichannel selection modes that enable or disable block transfers in each of the channels
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- Support for external generation of clock signals and frame-synchronization signals
- A programmable sample rate generator for internal generation and control of clock signals and frame-synchronization signals
- Programmable polarity for frame-synchronization pulses and clock signals
- Direct interface to:
 - T1/E1 framers
 - IOM-2 compliant devices
 - AC97-compliant devices (the necessary multiphase frame capability is provided)
 - I2S compliant devices
 - SPI devices
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits

NOTE: A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP documentation. Elsewhere, *word* is used to describe a 16-bit value.

- μ -law and A-law companding
- The option of transmitting/receiving 8-bit data with the LSB first
- Status bits for flagging exception/error conditions

- ABIS mode is not supported.

20.1.2 McBSP Pins/Signals

[Table 20-1](#) describes the McBSP interface pins and some internal signals.

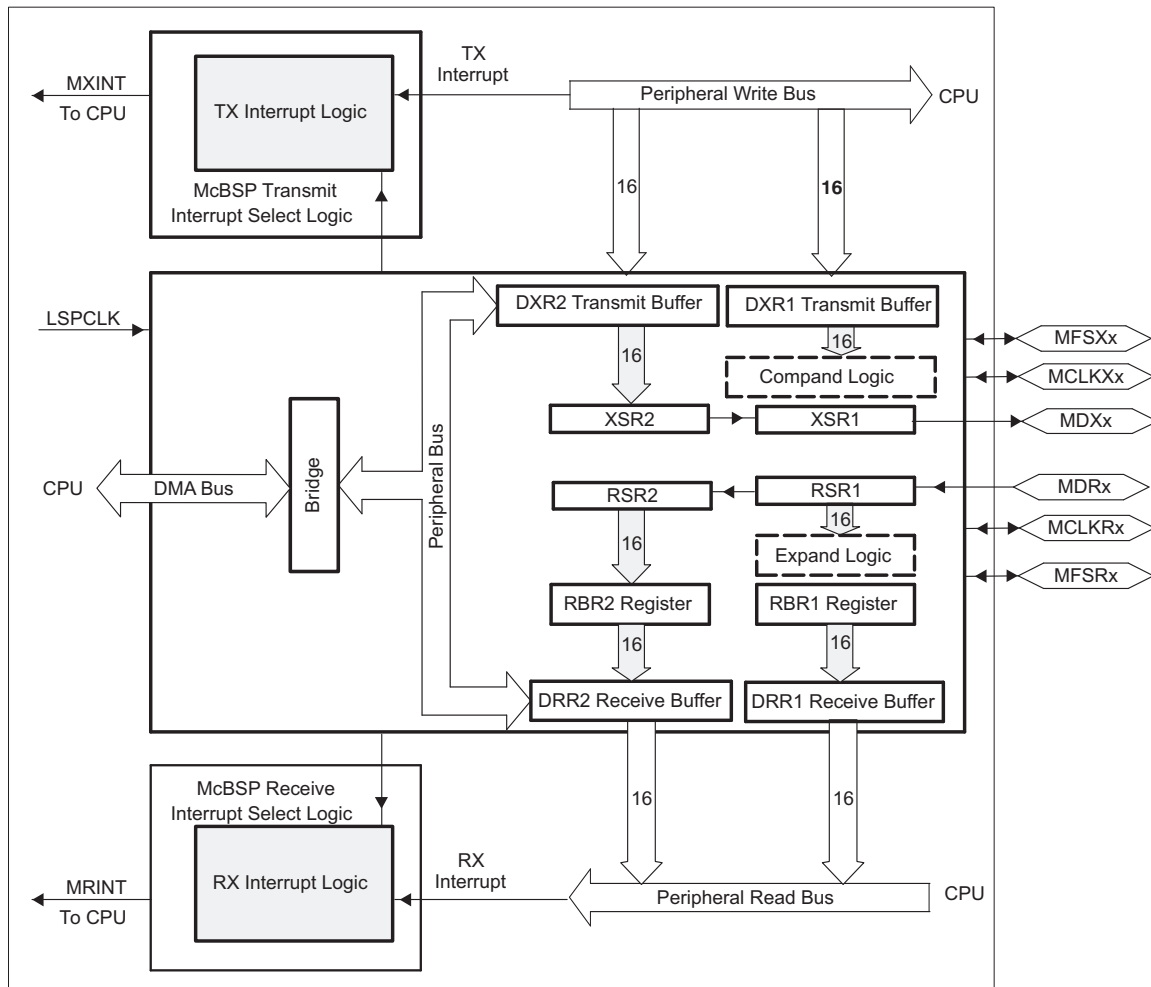
Table 20-1. McBSP Interface Pins/Signals

McBSP-A Pin	McBSP-B Pin	Type	Description
MCLKRA	MCLKRB	I/O	Supplying or reflecting the receive clock; supplying the input clock of the sample rate generator
MCLKXA	MCLKXB	I/O	Supplying or reflecting the transmit clock; supplying the input clock of the sample rate generator
MDRA	MDRB	I	Serial data receive pin
MDXA	MDXB	O	Serial data transmit pin
MFSRA	MFSRB	I/O	Supplying or reflecting the receive frame-sync signal; controlling sample rate generator synchronization for the case when GSYNC = 1 (see Section 20.4.3)
MFSXA	MFSXB	I/O	Supplying or reflecting the transmit frame-sync signal
CPU Interrupt Signals			
MRINT			Receive interrupt to CPU
MXINT			Transmit interrupt to CPU
DMA Events			
REVT			Receive synchronization event to DMA
XEVT			Transmit synchronization event to DMA

20.1.2.1 McBSP Generic Block Diagram

The McBSP consists of a data-flow path and a control path connected to external devices by six pins as shown in Figure 20-1. The figure and the text in this section use generic pin names.

Figure 20-1. Conceptual Block Diagram of the McBSP



A Not available in all devices. See the device-specific data sheet

20.2 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.

20.3 McBSP Operation

This section addresses the following topics:

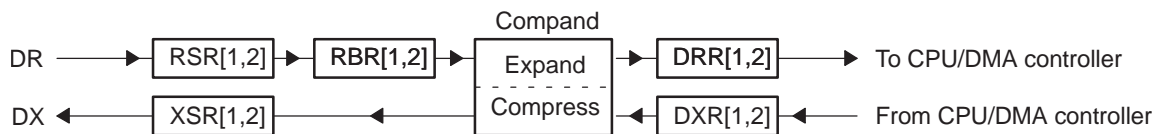
- Data transfer process
- Companding (compressing and expanding) data
- Clocking and framing data

- Frame phases
- McBSP reception
- McBSP transmission
- Interrupts and DMA events generated by McBSPs

20.3.1 Data Transfer Process of McBSPs

Figure 20-2 shows a diagram of the McBSP data transfer paths. The McBSP receive operation is triple-buffered, and transmit operation is double-buffered. The use of registers varies, depending on whether the defined length of each serial word is 16 bits.

Figure 20-2. McBSP Data Transfer Paths



20.3.1.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to receive buffer register 1 (RBR1) if RBR1 is not full with previous data. RBR1 is then copied to data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU or the DMA controller. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see [Section 20.3.5](#).

Transmit data is written by the CPU or the DMA controller to data transmit register 1 (DXR1). If there is no previous data in transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see [Section 20.3.6](#).

20.3.1.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted first into RSR2 and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU or the DMA controller. The CPU or the DMA controller must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see [Section 20.3.5](#).

For transmission, the CPU or the DMA controller must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see [Section 20.3.6](#).

20.3.2 Companding (Compressing and Expanding) Data

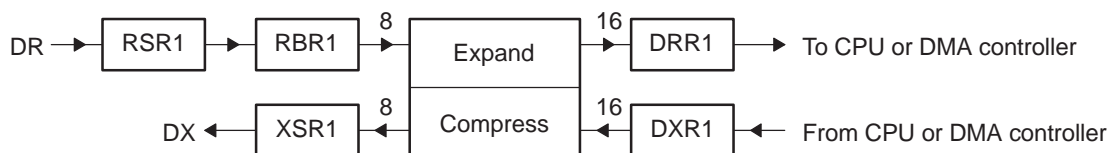
Companding (COMpressing and exPANDING) hardware allows compression and expansion of data in either μ -law or A-law format. The companding standard employed in the United States and Japan is μ -law. The European companding standard is referred to as A-law. The specifications for μ -law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ -law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ -law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 20-3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ -law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to two's-complement format.

Figure 20-3. Companding Processes

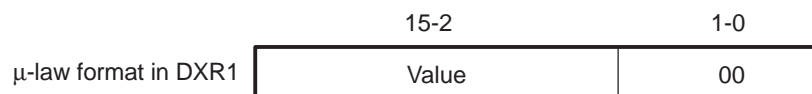


20.3.2.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

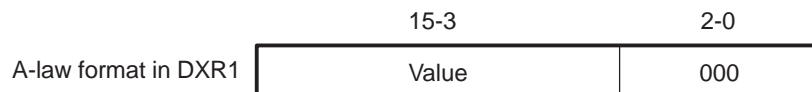
For transmission using μ -law compression, the 14 data bits must be left-justified in DXR1 and that the remaining two low-order bits are filled with 0s as shown in Figure 20-4.

Figure 20-4. μ -Law Transmit Data Companding Format



For transmission using A-law compression, the 13 data bits must be left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 20-5.

Figure 20-5. A-Law Transmit Data Companding Format



20.3.2.2 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ -law or A-law format
- Convert μ -law or A-law to the linear format
- Observe the quantization effects in companding by transmitting linear data and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

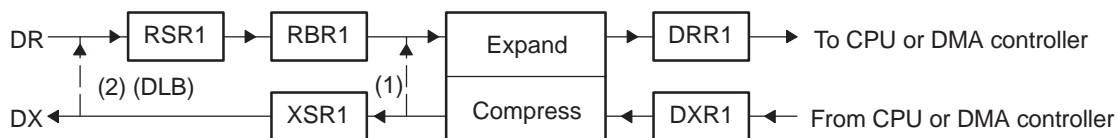
Figure 20-6 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.

The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow. DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using μ -law or A-law).

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 20-6. Two Methods by Which the McBSP Can Compand Internal Data



20.3.2.3 Reversing Bit Order: Option to Transfer LSB First

Generally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

20.3.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

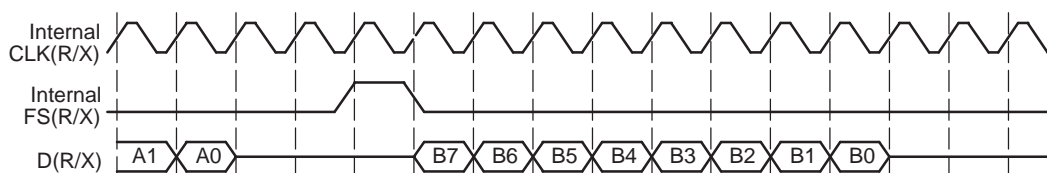
20.3.3.1 Clocking

Data is shifted one bit at a time from the DR pin to the RSR(s) or from the XSR(s) to the DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the DR pin to the RSR(s). The transmit clock signal (CLKX) controls bit transfers from the XSR(s) to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

In the example in Figure 20-7, the clock signal controls the timing of each bit transfer on the pin.

Figure 20-7. Example - Clock Signal Control of Bit Transfer Timing



NOTE: The McBSP cannot operate at a frequency faster than $\frac{1}{2}$ the LSPCLK frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV) (i.e., be certain that $\text{CLKX or CLKR} \leq \text{LSPCLK}/2$).

20.3.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a serial word. You can define how many bits are in a word.

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In the example in [Figure 20-7](#), an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

20.3.3.3 Frames and Frame Synchronization

One or more words are transferred in a group called a frame. You can define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the receive frame-synchronization (FSR) signal initiate frame transfers on DR. Pulses on the transmit frame-sync (FSX) signal initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In the example in [Figure 20-7](#), a one-word frame is transferred when a frame-synchronization pulse occurs.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

20.3.3.4 Generating Transmit and Receive Interrupts

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-synchronization pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

20.3.3.4.1 Detecting Frame-Synchronization Pulses, Even in Reset State

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

20.3.3.5 Ignoring Frame-Synchronization Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-synchronization pulses, clear the appropriate frame-synchronization ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-synchronization pulses until the desired frame length or number of words is reached, set the appropriate frame-synchronization ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-synchronization pulses, see one of the following topics:

- *Unexpected Receive Frame-Synchronization Pulse* (see [Section 20.5.3](#))
- *Unexpected Transmit Frame-Synchronization Pulse* (see [Section 20.5.6](#))

You can also use the frame-synchronization ignore function for data packing (for more details, see [Section 20.11.2](#)).

20.3.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown by Example 1.

Example 1: McBSP Frame Frequency

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame-Sync Pulses}}$$

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

20.3.3.7 Maximum Frame Frequency

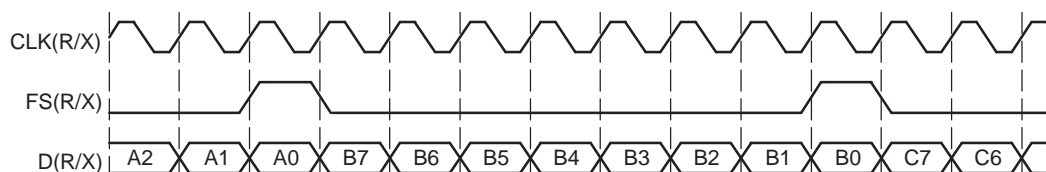
The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown by Example 2.

Example 2: McBSP Maximum Frame Frequency

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

[Figure 20-8](#) shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 20-8. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in this figure, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-synchronization pulses. Data is clocked into the receiver or clocked out of the transmitter during every clock cycle.

NOTE: For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see [Section 20.9.12](#).

20.3.4 Frame Phases

The McBSP allows you to configure each frame to contain one or two phases. The number of words and the number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, you might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of 10 words of 8 bits each. This configuration permits you to compose frames for custom applications or, in general, to maximize the efficiency of data transfers.

20.3.4.1 Number of Phases, Words, and Bits Per Frame

Table 20-2 shows which bit-fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) determine the number of phases per frame, the number of words per frame, and number of bits per word for each phase, for the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Table 20-2. Register Bits That Determine the Number of Phases, Words, and Bits

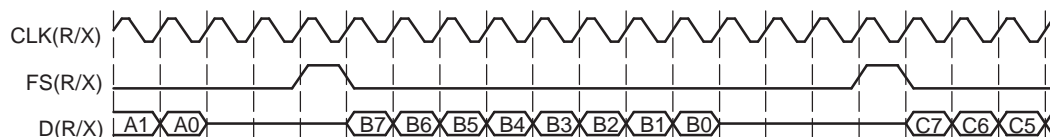
Operation	Number of Phases	Words per Frame Set With	Bits per Word Set With
Reception	1 (RPHASE = 0)	RFRLN1	RWDLEN1
Reception	2 (RPHASE = 1)	RFRLN1 and RFRLN2	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLN1 and XFRLN2	XWDLEN1 for phase 1 XWDLEN2 for phase 2

20.3.4.2 Single-Phase Frame Example

Figure 20-9 shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for one data bit delay, the data on the DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLN1 = 0b: 1 word per frame
- (R/X)WDLEN1 = 000b: 8-bit word length
- (R/X)FRLN2 and (R/X)WDLEN2 are ignored
- CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- FS(R/X)P = 0: Active-high frame-synchronization signals
- (R/X)DATDLY = 01b: 1-bit data delay

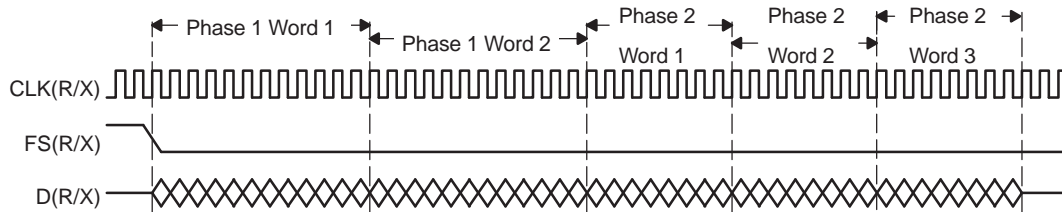
Figure 20-9. Single-Phase Frame for a McBSP Data Transfer



20.3.4.3 Dual-Phase Frame Example

Figure 20-10 shows an example of a frame where the first phase consists of two words of 12 bits each, followed by a second phase of three words of 8 bits each. The entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

Figure 20-10. Dual-Phase Frame for a McBSP Data Transfer

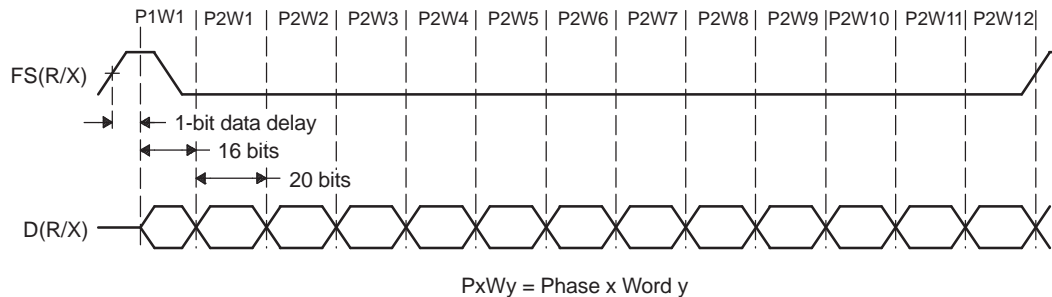


A XRDY gets asserted once per phase. So, if there are 2 phases, XRDY gets asserted twice (once per phase).

20.3.4.4 Implementing the AC97 Standard With a Dual-Phase Frame

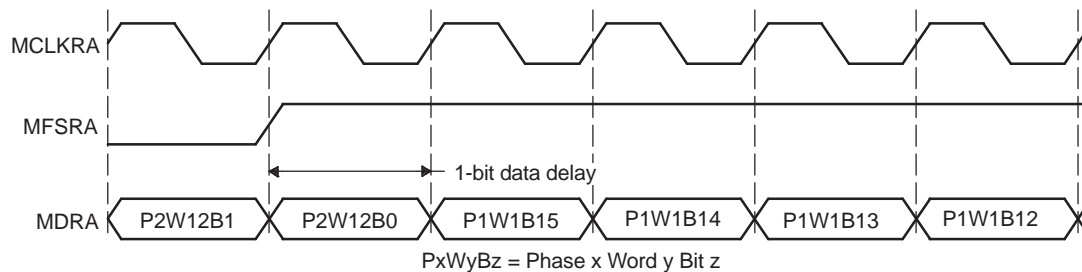
Figure 20-11 shows an example of the Audio Codec '97 (AC97) standard, which uses the dual-phase frame feature. Notice that words, not individual bits, are shown on the D(R/X) signal. The first phase (P1) consists of a single 16-bit word. The second phase (P2) consists of twelve 20-bit words. The phase configurations are listed after the figure.

Figure 20-11. Implementing the AC97 Standard With a Dual-Phase Frame



- (R/X)PHASE = 1: Dual-phase frame
- (R/X)FRLN1 = 0000000b: 1 word in phase 1
- (R/X)WDLEN1 = 010b: 16 bits per word in phase 1
- (R/X)FRLN2 = 0001011b: 12 words in phase 2
- (R/X)WDLEN2 = 011b: 20 bits per word in phase 2
- CLKRP/CLKXP = 0: Receive data sampled on falling edge of internal CLKR / transmit data clocked on rising edge of internal CLKX
- FSRP/FSXP = 0: Active-high frame-sync signal
- (R/X)DATDLY = 01b: Data delay of 1 clock cycle (1-bit data delay)

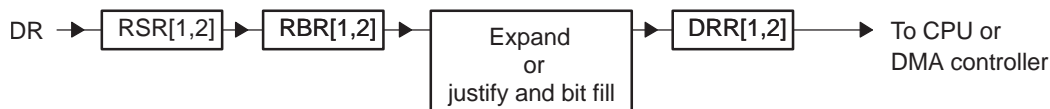
Figure 20-12 shows the timing of an AC97-standard data transfer near frame synchronization. In this figure, individual bits are shown on D(R/X). Specifically, the figure shows the last two bits of phase 2 of one frame and the first four bits of phase 1 of the next frame. Regardless of the data delay, data transfers can occur without gaps. The first bit of the second frame (P1W1B15) immediately follows the last bit of the first frame (P2W12B0). Because a 1-bit data delay has been chosen, the transition on the frame-sync signal can occur when P2W12B0 is transferred.

Figure 20-12. Timing of an AC97-Standard Data Transfer Near Frame Synchronization


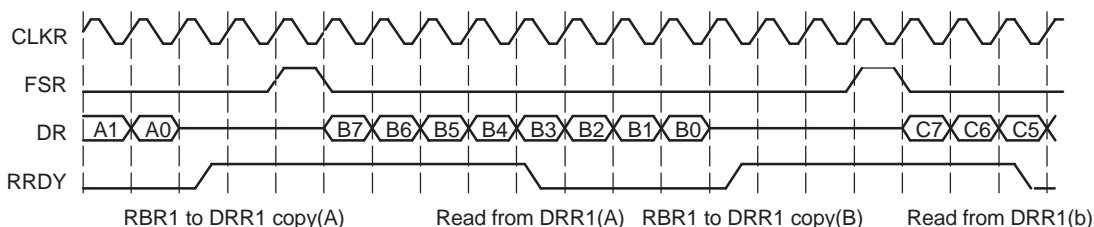
20.3.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see [Section 20.8](#).

[Figure 20-13](#) and [Figure 20-14](#) show how reception occurs in the McBSP. [Figure 20-13](#) shows the physical path for the data. [Figure 20-14](#) is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 20-13. McBSP Reception Physical Data Path


- A RSR[1,2]: Receive shift registers 1 and 2
- B RBR[1,2]: Receive buffer registers 1 and 2
- C DRR[1,2]: Data receive registers 1 and 2

Figure 20-14. McBSP Reception Signal Activity


- A CLKR: Internal receive clock
- B FSR: Internal receive frame-synchronization signal
- C DR: Data on DR pin
- D RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the DR pin to the CPU or to the DMA controller:

1. The McBSP waits for a receive frame-synchronization pulse on internal FSR.
2. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.
In the preceding timing diagram, a 1-bit data delay is selected.
3. The McBSP accepts data bits on the DR pin and shifts them into the receive shift register(s).
If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used and RSR2 contains the most significant bits. For details on choosing a word length, see [Section 20.8.8, Set the Receive Word Length\(s\)](#).
4. When a full word is received, the McBSP copies the contents of the receive shift register(s) to the receive buffer register(s), provided that RBR1 is not full with previous data.
If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used and RBR2 contains the most significant bits.

- The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. This indicates that received data is ready to be read by the CPU or the DMA controller.
If the word length is 16 bits or smaller, only DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used and DRR2 contains the most significant bits.
If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.
- The CPU or the DMA controller reads the data from the data receive register(s). When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

NOTE: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

When activity is not properly timed, errors can occur. See the following topics for more details:

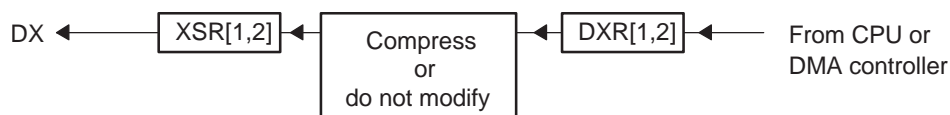
- Overflow in the Receiver* (see [Section 20.5.2](#))
- Unexpected Receive Frame-Synchronization Pulse* (see [Section 20.5.3](#))

20.3.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see [Section 20.9](#).

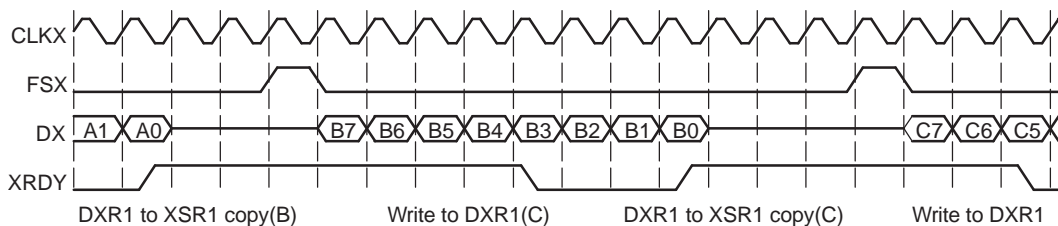
[Figure 20-15](#) and [Figure 20-16](#) show how transmission occurs in the McBSP. [Figure 20-15](#) shows the physical path for the data. [Figure 20-16](#) is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 20-15. McBSP Transmission Physical Data Path



- A XSR[1,2]: Transmit shift registers 1 and 2
B DXR[1,2]: Data transmit registers 1 and 2

Figure 20-16. McBSP Transmission Signal Activity



- A CLKX: Internal transmit clock
B FSX: Internal transmit frame-synchronization signal
C DX: Data on DX pin
D XRDY: Status of transmitter ready bit (high is 1)

- The CPU or the DMA controller writes data to the data transmit register(s). When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data.

If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used and DXR2 contains the most significant bits. For details on choosing a word length, see [Section 20.9.8](#).

NOTE: If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

2. When new data arrives in DXR1, the McBSP copies the content of the data transmit register(s) to the transmit shift register(s). In addition, the transmit ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU or the DMA controller.
If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used and XSR2 contains the most significant bits.
If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ -law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.
3. The McBSP waits for a transmit frame-synchronization pulse on internal FSX.
4. When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.
In the preceding timing diagram (Figure 20-16), a 1-bit data delay is selected.
5. The McBSP shifts data bits from the transmit shift register(s) to the DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

- *Overwrite in the Transmitter* (Section 20.5.4)
- *Underflow in the Transmitter* (Section 20.5.5)
- *Unexpected Transmit Frame-Synchronization Pulse* (Section 20.5.6)

20.3.7 Interrupts and DMA Events Generated by a McBSP

The McBSP sends notification of important events to the CPU and DMA via the internal signals shown in Table 20-3.

Table 20-3. Interrupts and DMA Events Generated by a McBSP

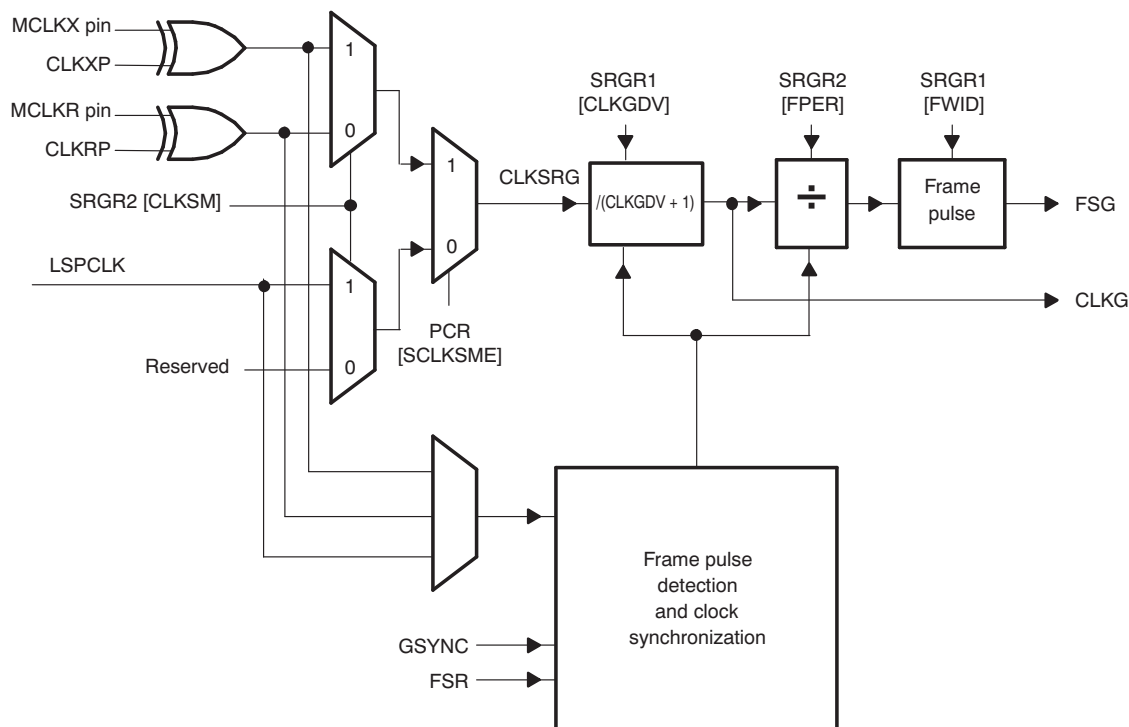
Internal Signal	Description
RINT	Receive interrupt The McBSP can send a receive interrupt request to CPU based upon a selected condition in the receiver of the McBSP (a condition selected by the RINTM bits of SPCR1).
XINT	Transmit interrupt The McBSP can send a transmit interrupt request to CPU based upon a selected condition in the transmitter of the McBSP (a condition selected by the XINTM bits of SPCR2).
REVT	Receive synchronization event An REVT signal is sent to the DMA when data has been received in the data receive registers (DRRs).
XEVT	Transmit synchronization event An XEVT signal is sent to the DMA when the data transmit registers (DXRs) are ready to accept the next serial word for transmission.

20.4 McBSP Sample Rate Generator

Each McBSP contains a sample rate generator (SRG) that can be programmed to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX. Figure 20-17 is a conceptual block diagram of the sample rate generator.

20.4.1 Block Diagram

Figure 20-17. Conceptual Block Diagram of the Sample Rate Generator



The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by the LSPCLK, or by an external pin (MCLKX or MCLKR). The source is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKXP of PCR or CLKRP of PCR).

The sample rate generator has a three-stage clock divider that gives CLKG and FSG programmability. The three stages provide:

- Clock divide-down. The source clock is divided according to the CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide-down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-pulse to the start of the next pulse.
- Frame-synchronization pulse-width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-synchronization pulse.

NOTE: The McBSP cannot operate at a frequency faster than $\frac{1}{2}$ the source clock frequency. Choose an input clock frequency and a CLKGDV value such that CLKG is less than or equal to $\frac{1}{2}$ the source clock frequency.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see [Section 20.4.4](#).

20.4.1.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG).

The effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in [Table 20-4](#). The digital loopback mode (described in [Section 20.8.4](#)) is selected with the DLB bit of SPCR1. The clock stop mode (described in [Section 20.7.2](#)) is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled (GRST = 1).

Table 20-4. Effects of DLB and CLKSTP on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 1 (Digital loopback mode enabled)	CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit.
CLKXM = 1	CLKSTP = 00b or 01b (Clock stop (SPI) mode disabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	CLKSTP = 10b or 11b (Clock stop (SPI) mode enabled)	The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal sample rate generator.

20.4.1.2 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the three sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see [Table 20-5](#)). When CLKSM = 1, the minimum divide down value in CLKGDV bits is 1. CLKGDV is described in [Section 20.4.1.4](#).

Table 20-5. Choosing an Input Clock for the Sample Rate Generator with the SCLKME and CLKSM Bits

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on MCLKR pin
1	1	Signal on MCLKX pin

20.4.1.3 Choosing a Polarity for the Input Clock

As shown in [Figure 20-18](#), when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in [Table 20-6](#).

Figure 20-18. Possible Inputs to the Sample Rate Generator and the Polarity Bits

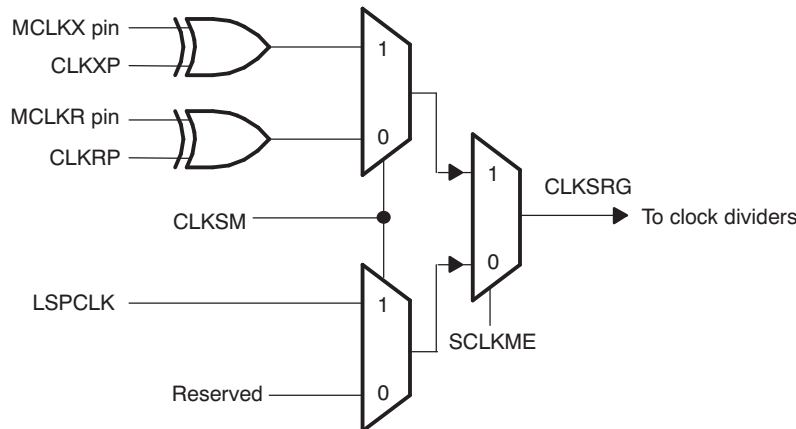


Table 20-6. Polarity Options for the Input to the Sample Rate Generator

Input Clock	Polarity Option	Effect
LSPCLK	Always positive polarity	Rising edge of CPU clock generates transitions on CLKG and FSG.
Signal on MCLKR pin	CLKRP = 0 in PCR	Falling edge on MCLKR pin generates transitions on CLKG and FSG.
	CLKRP = 1 in PCR	Rising edge on MCLKR pin generates transitions on CLKG and FSG.
Signal on MCLKX pin	CLKXP = 0 in PCR	Rising edge on MCLKX pin generates transitions on CLKG and FSG.
	CLKXP = 1 in PCR	Falling edge on MCLKX pin generates transitions on CLKG and FSG.

20.4.1.4 Choosing a Frequency for the Output Clock (CLKG)

The input clock (LSPCLK or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see [Figure 20-1](#)) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by the equation.

Equation 1: CLKG Frequency

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

20.4.1.4.1 CLKG Frequency

Thus, the input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

20.4.1.5 Keeping CLKG Synchronized to External MCLKR

When the MCLKR pin is used to drive the sample rate generator (see [Section 20.4.1.2](#)), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock. Note that this feature is available only when the MCLKR pin is used to feed the external clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about synchronization, see [Section 20.4.3](#).

20.4.2 Frame Synchronization Generation in the Sample Rate Generator

The sample rate generator can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the FSX pin.
- FSGM = 1 in SRGR2: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-synchronization pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled (GRST = 1) and the frame-synchronization logic in the sample rate generator must be enabled (FRST = 1).

20.4.2.1 Choosing the Width of the Frame-Synchronization Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

20.4.2.2 Controlling the Period Between the Starting Edges of Frame-Synchronization Pulses on FSG

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-synchronization period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-synchronization period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

20.4.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see [Section 20.4.1.2](#) on page [Section 20.4.1.2](#)), the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

See [Section 20.4.3](#) for more details about synchronization.

20.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the MCLKR or MCLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.

- The FPER bits of SRGR2 are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-synchronization period on FSG is determined by FPER.

20.4.3.1 Operating the Transmitter Synchronously with the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2 and FSXM = 1 in PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR).

20.4.3.2 Synchronization Examples

Figure 20-19 and Figure 20-20 show the clock and frame-synchronization operation with various polarities of CLKR and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is one CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. The second figure has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

Figure 20-19. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

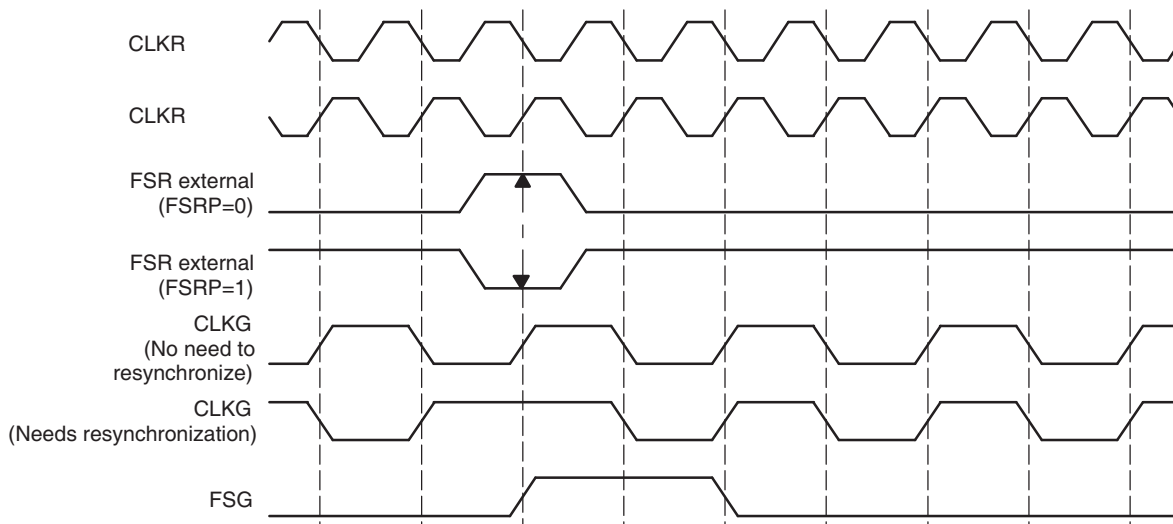
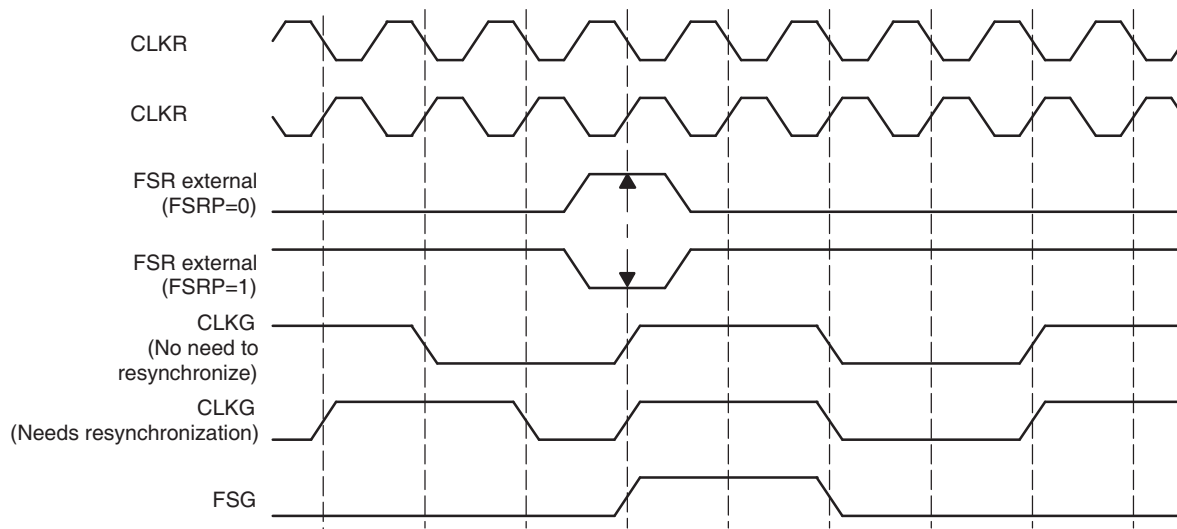


Figure 20-20. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3


20.4.4 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

Step 1. Place the McBSP/sample rate generator in reset.

During a DSP reset, the sample rate generator, the receiver, and the transmitter reset bits (GRST, RRST, and XRST) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making GRST = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver (RRST = 0 in SPCR1) and reset the transmitter (XRST = 0 in SPCR2).

If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If GRST = 0 due to program code, CLKG and FSG are driven low (inactive).

Step 2. Program the registers that affect the sample rate generator.

Program the sample rate generator registers (SRGR1 and SRGR2) as required for your application. If necessary, other control registers can be loaded with desired values, provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

After the sample rate generator registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.

Step 3. Enable the sample rate generator (take it out of reset).

In SPCR2, make GRST = 1 to enable the sample rate generator.

After the sample rate generator is enabled, wait two CLKG cycles for the sample rate generator logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to the CLKG Frequency equation below.

Table 20-7. Input Clock Selection for Sample Rate Generator

SCLKME	CLKSM	Input Clock for Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on MCLKR pin
1	1	Signal on MCLKX pin

Step 4. If necessary, enable the receiver and/or the transmitter.

If necessary, remove the receiver and/or transmitter from reset by setting RRST and/or XRST = 1.

Step 5. If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1,2] is loaded with data), set GRST = 1 in SPCR2 if an internally generated frame-synchronization pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

Equation 2: CLKG Frequency

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2 in one of the configurations shown in [Table 20-7](#).

20.5 McBSP Exception/Error Conditions

This chapter describes exception/error conditions and how to handle them.

20.5.1 Types of Errors

There are five serial port events that can constitute a system error:

- Receiver overrun (RFULL = 1)
This occurs when DRR1 has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBR(s) to the DRR(s) and the RSR(s) are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that can arrive at this time on DR replaces the contents of the RSR(s), and the previous word is lost. The RSRs continue to be overwritten as long as new data arrives on DR and DRR1 is not read. For more details about overrun in the receiver, see [Section 20.5.2](#).
- Unexpected receive frame-synchronization pulse (RSYNCERR = 1)
This occurs during reception when RFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBR(s) from the RSR(s) since the last RBR-to-DRR copy, this new data in the RBR(s) is lost. This is because no RBR-to-DRR copy occurs; the reception has been restarted. For more details about receive frame-synchronization errors, see [Section 20.5.3](#).
- Transmitter data overwrite
This occurs when the CPU or DMA controller overwrites data in the DXR(s) before the data is copied to the XSR(s). The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see [Section 20.5.4](#).
- Transmitter underflow (XEMPTY = 0)
If a new frame-synchronization signal arrives before new data is loaded into DXR1, the previous data in the DXR(s) is sent again. This procedure continues for every new frame-synchronization pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see [Section 20.5.5](#).
- Unexpected transmit frame-synchronization pulse (XSYNCERR = 1)
This occurs during transmission when XFIG = 0 and an unexpected frame-synchronization pulse occurs. An unexpected frame-synchronization pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the DXR(s) since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more details about transmit frame-synchronization errors, see [Section 20.5.6](#).

20.5.2 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1. DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 1).
2. RBR1 is full and an RBR-to-DRR copy has not occurred.
3. RSR1 is full and an RSR1-to-RBR copy has not occurred.

As described in the [Section 20.3.5](#), data arriving on DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. Until RRDY = 0, the next RBR-to-DRR copy does not take place, and the data is held in the RSR(s). New data arriving on the DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

NOTE: If both DRRs are needed (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

After the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

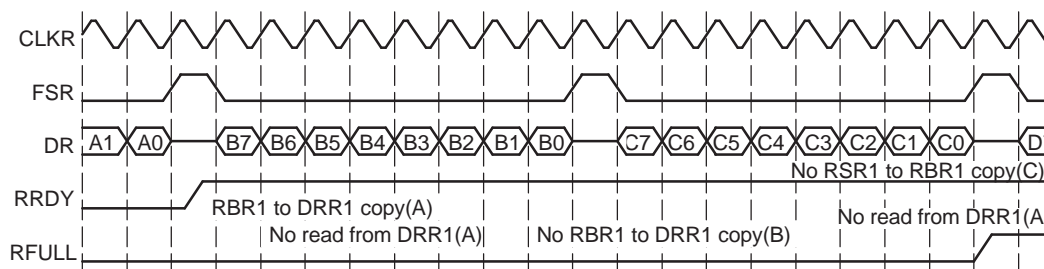
- The CPU or DMA controller reads DRR1.
- The receiver is reset individually (RRST = 0) or as part of a device reset.

Another frame-synchronization pulse is required to restart the receiver.

20.5.2.1 Example of Overrun Condition

[Figure 20-21](#) shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word ©) arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If DRR1 is not read in time, the next word can overwrite D.

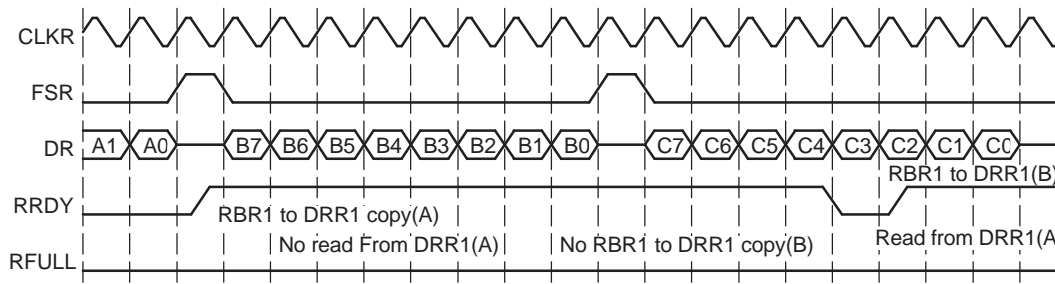
Figure 20-21. Overrun in the McBSP Receiver



20.5.2.2 Example of Preventing Overrun Condition

[Figure 20-22](#) shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 at least 2.5 cycles before the next serial word ©) is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of word B occurs before receiver attempts to transfer word C from RSR1 to RBR1.

Figure 20-22. Overrun Prevented in the McBSP Receiver



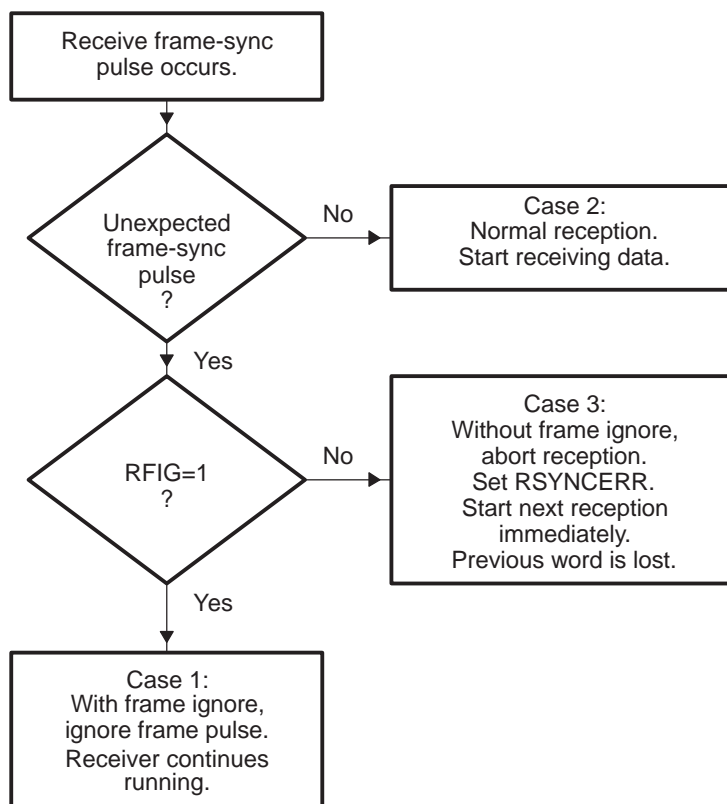
20.5.3 Unexpected Receive Frame-Synchronization Pulse

Section 20.5.3.1 shows how the McBSP responds to any receive frame-synchronization pulses, including an unexpected pulse. Section 20.5.3.2 and Section 20.5.3.3 show an examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

20.5.3.1 Possible Responses to Receive Frame-Synchronization Pulses

Figure 20-23 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The figure assumes that the receiver has been started (RRST = 1 in SPCR1). Case 3 in the figure is the case in which an error occurs.

Figure 20-23. Possible Responses to Receive Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-synchronization pulses are ignored, and the reception continues.
- Case 2: Normal serial port reception. Reception continues normally because the frame-synchronization

pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:

- The FSR pulse is the first after the receiver is enabled (RRST = 1 in SPCR1).
- The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.
- The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATADELAY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

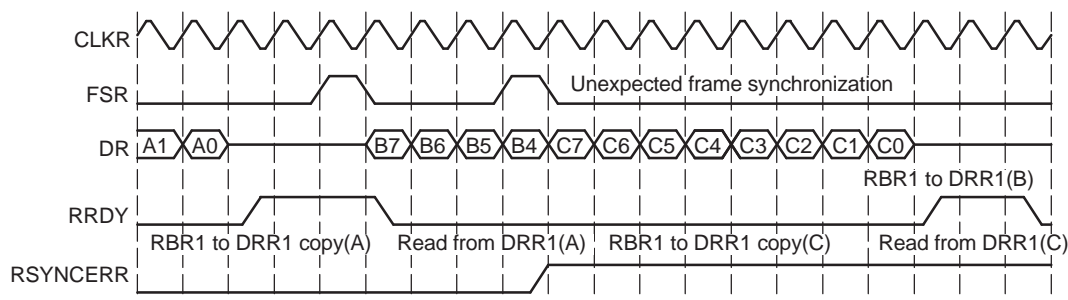
If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of receive frame-synchronization errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

20.5.3.2 Example of Unexpected Receive Frame-Synchronization Pulse

Figure 20-30 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-synchronization pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

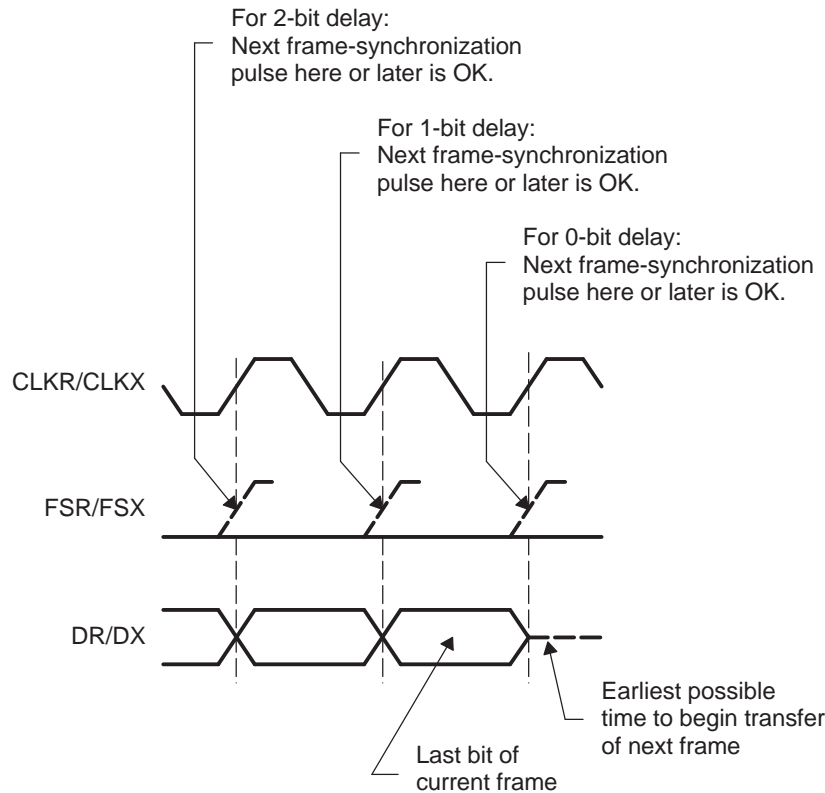
Figure 20-24. An Unexpected Frame-Synchronization Pulse During a McBSP Reception



20.5.3.3 Preventing Unexpected Receive Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 MCLKR cycles, depending on the value in the RDATADELAY bits of RCR2. For each possible data delay, Figure 20-25 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 20-25. Proper Positioning of Frame-Synchronization Pulses



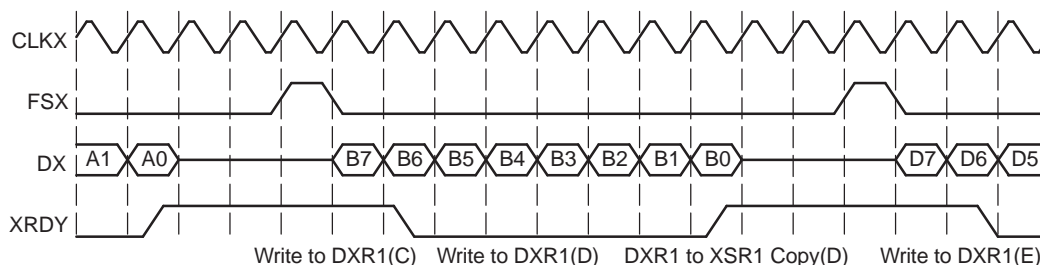
20.5.4 Overwrite in the Transmitter

As described in the section on McBSP transmission (page [Section 20.3.6](#)), the transmitter must copy the data previously written to the DXR(s) by the CPU or DMA controller into the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

20.5.4.1 Example of Overwrite Condition

Figure 20-26 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

Figure 20-26. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted



20.5.4.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- Poll for XRDY = 1 in SPCR2 before writing to the DXR(s). XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.

- Wait for a transmit interrupt (XINT) before writing to the DXR(s). When XINTM = 00b in SPCR2, the transmitter sends XINT to the CPU each time XRDY is set.

You can prevent DMA overwrites by synchronizing DMA transfers to the transmit synchronization event XEVT. The transmitter sends an XEVT signal each time XRDY is set.

20.5.5 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ($\overline{\text{XEMPTY}} = 0$):

- DXR1 has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the DX pin.
- The transmitter is reset (by forcing XRST = 0 in SPCR2, or by a device reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-synchronization signal until a new value is loaded into DXR1 by the CPU or the DMA controller.

NOTE: If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

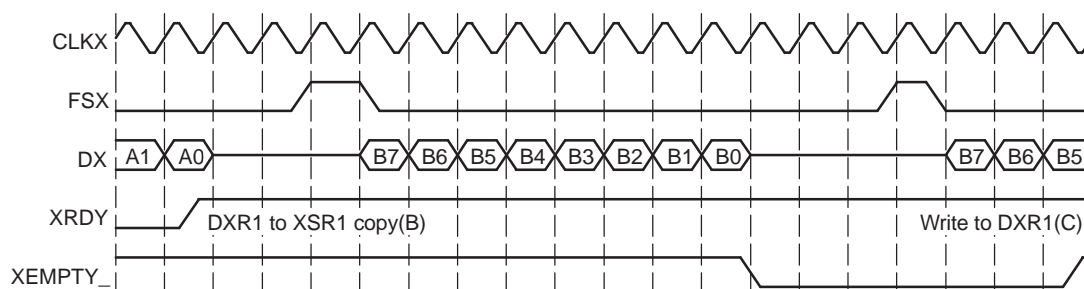
XEMPTY is deactivated ($\overline{\text{XEMPTY}} = 1$) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on DX.

When the transmitter is taken out of reset (XRST = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty ($\overline{\text{XEMPTY}} = 0$) state. If DXR1 is loaded by the CPU or the DMA controller before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before DXR1 is loaded, zeros are output on DX.

20.5.5.1 Example of the Underflow Condition

Figure 20-27 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-synchronization pulse. Thus, B is again transmitted on DX.

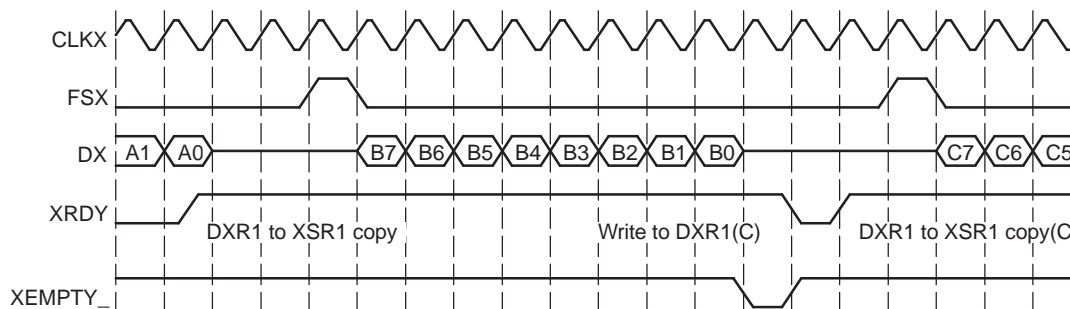
Figure 20-27. Underflow During McBSP Transmission



20.5.5.2 Example of Preventing Underflow Condition

Figure 20-28 shows the case of writing to DXR1 just before an underflow condition would otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-synchronization pulse. As a result, there is no underflow; B is not transmitted twice.

Figure 20-28. Underflow Prevented in the McBSP Transmitter



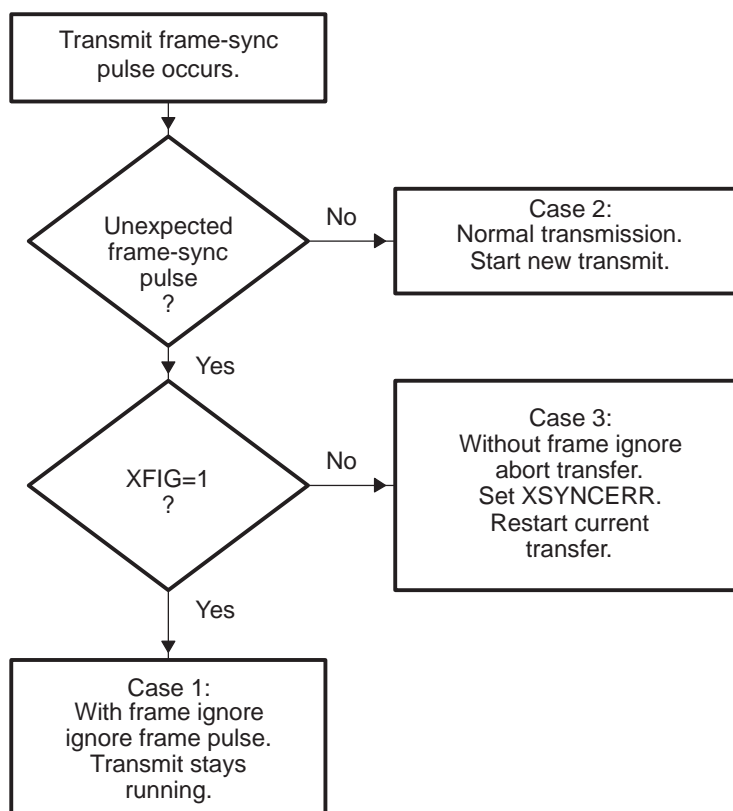
20.5.6 Unexpected Transmit Frame-Synchronization Pulse

Section 20.5.6.1 shows how the McBSP responds to any transmit frame-synchronization pulses, including an unexpected pulse. Section 20.5.6.2 and Section 20.5.6.3 show examples of a frame-synchronization error and an example of how to prevent such an error, respectively.

20.5.6.1 Possible Responses to Transmit Frame-Synchronization Pulses

Figure 20-29 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization pulses. The figure assumes that the transmitter has been started (XRST = 1 in SPCR2). Case 3 in the figure is the case in which an error occurs.

Figure 20-29. Possible Responses to Transmit Frame-Synchronization Pulses



Any one of three cases can occur:

- Case 1: Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-synchronization pulses are ignored, and the transmission continues.
- Case 2: Normal serial port transmission. Transmission continues normally because the frame-

synchronization pulse is not unexpected. There are two possible reasons why a transmit operations might *not* be in progress when the pulse occurs:

This FSX pulse is the first after the transmitter is enabled (XRST = 1).

The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- Case 3: Unexpected transmit frame synchronization with XFIG = 0 (frame-synchronization pulses not ignored). Unexpected frame-synchronization pulses can originate from an external source or from the internal sample rate generator.

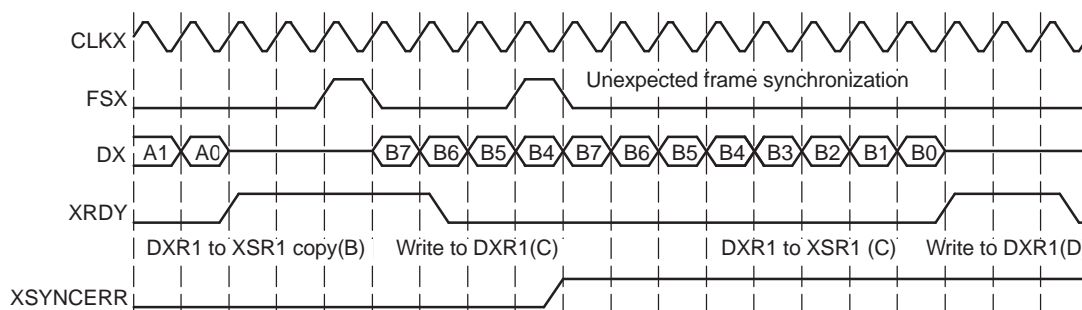
If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of frame-synchronization errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

20.5.6.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Section 20.5.3.2 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets. When the unexpected frame-synchronization pulse occurs, the XSYNCERR bit is set and the transmission of data B is restarted because no new data has been passed to XSR1 yet. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

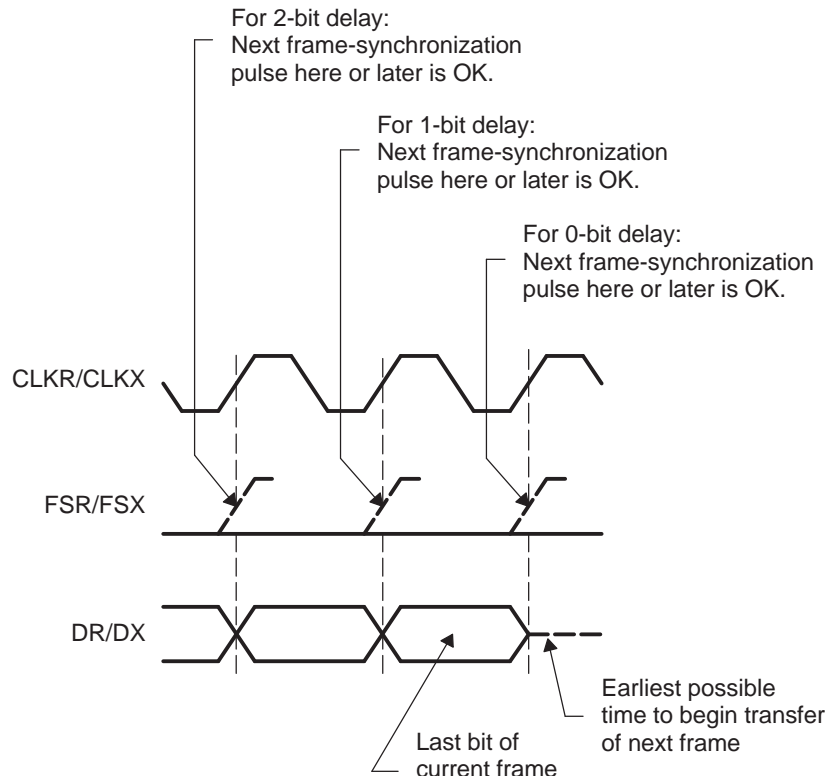
Figure 20-30. An Unexpected Frame-Synchronization Pulse During a McBSP Transmission



20.5.6.3 Preventing Unexpected Transmit Frame-Synchronization Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 20-31 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 20-31. Proper Positioning of Frame-Synchronization Pulses



20.6 Multichannel Selection Modes

This section discusses the multichannel selection modes for the McBSP.

20.6.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels (see [Table 20-8](#) through [Table 20-10](#)) :

- It is possible to have two receive partitions (A & B) and 8 transmit partitions (A – H).
- McBSP can transmit/receive on selected channels.
- Each channel partition has a dedicated channel-enable register. Each bit controls whether data flow is allowed or prevented in one of the channels assigned to that partition.
- There are three transmit multichannel modes and one receive multichannel mode.

Table 20-8. Block - Channel Assignment

Block	Channels
0	0 - 15
1	16 - 31
2	32 - 47
3	48 - 63
4	64 - 79
5	80 - 95
6	96 - 111
7	112 - 127

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode (described in [Section 20.6.4](#)), you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode (described in [Section 20.6.5](#)), blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

Table 20-9. 2-Partition Mode

Partition	Blocks
A	0 or 2 or 4 or 6
B	1 or 3 or 5 or 7

Table 20-10. 8-Partition mode

Partition	Blocks	Channels
A	0	0 - 15
B	1	16 - 31
C	2	32 - 47
D	3	48 - 63
E	4	64 - 79
F	5	80 - 95
G	6	96 - 111
H	7	112 - 127

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A-H).

20.6.2 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode (described in [Section 20.6.6](#)) and three transmit multichannel selection modes (described in [Section 20.6.7](#)).

20.6.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (in RFLEN1/XFLEN1) that includes the highest-numbered channel to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFLEN1 = 39). If XFLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

20.6.4 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions (described in [Section 20.6.5](#)). If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

20.6.4.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up to 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

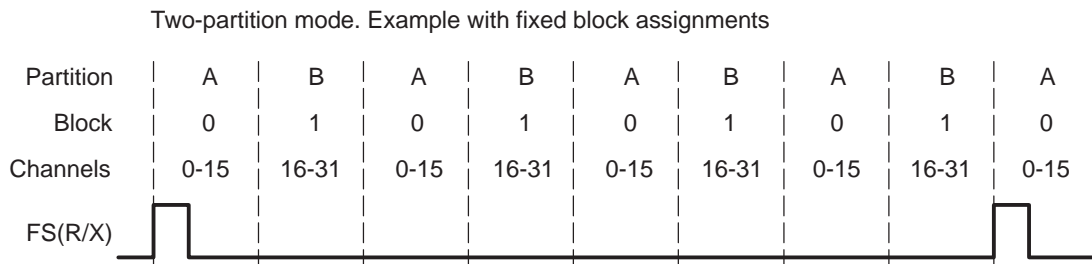
- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in [Section 20.6.6](#)), the channels in this partition are controlled by receive channel enable register A (RCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (RCERB).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes (described in [Section 20.6.7](#)), the channels in this partition are controlled by transmit channel enable register A (XCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (XCERB).

[Figure 20-32](#) shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 20-32. Alternating Between the Channels of Partition A and the Channels of Partition B



As explained in [Section 20.6.4.2](#), you can dynamically change which blocks of channels are assigned to the partitions.

20.6.4.2 Reassigning Blocks During Reception/Transmission

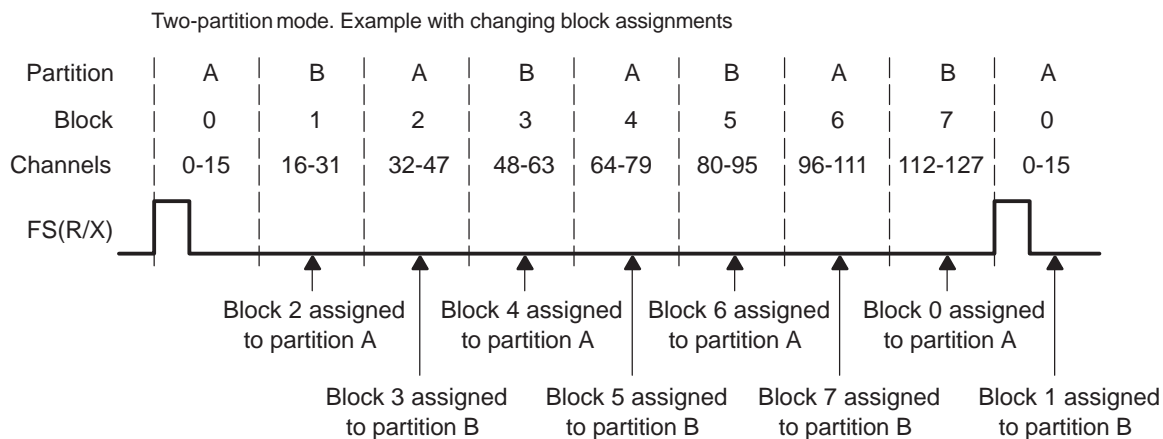
If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, its associated block assignment bits cannot be modified and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you can modify neither (R/X)PABLK to assign different channels to partition A nor (R/X)CERA to change the channel configuration for partition A.

Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition. See [Section 20.6.8](#).

[Figure 20-33](#) shows an example of reassigning channels throughout a data transfer. In response to a frame-synchronization pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

Figure 20-33. Reassigning Channel Blocks Throughout a McBSP Data Transfer



20.6.5 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (described in [Section 20.6.4](#)). If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-synchronization pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in [Table 20-11](#) and [Table 20-12](#). These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 20-11. Receive Channel Assignment and Control With Eight Receive Partitions

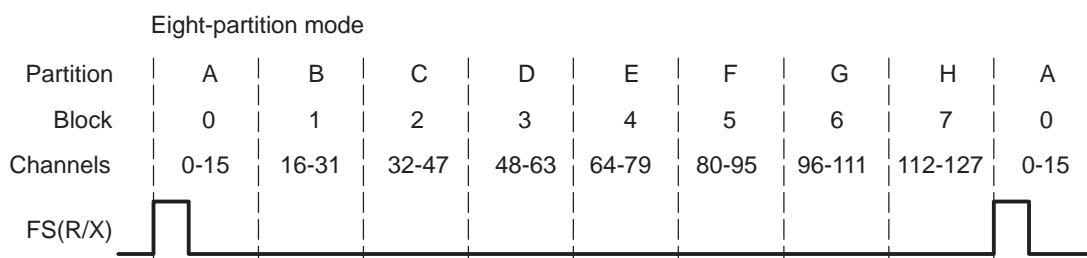
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERA
B	Block 1: channels 16 through 31	RCERB
C	Block 2: channels 32 through 47	RCERC
D	Block 3: channels 48 through 63	RCERD
E	Block 4: channels 64 through 79	RCERE
F	Block 5: channels 80 through 95	RCERF
G	Block 6: channels 96 through 111	RCERG
H	Block 7: channels 112 through 127	RCERH

Table 20-12. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCERA
B	Block 1: channels 16 through 31	XCERB
C	Block 2: channels 32 through 47	XCERC
D	Block 3: channels 48 through 63	XCERD
E	Block 4: channels 64 through 79	XCERE
F	Block 5: channels 80 through 95	XCERF
G	Block 6: channels 96 through 111	XCERG
H	Block 7: channels 112 through 127	XCERH

Figure 20-34 shows an example of the McBSP using the 8-partition mode. In response to a frame-synchronization pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 20-34. McBSP Data Transfer in the 8-Partition Mode



20.6.6 Receive Multichannel Selection Mode

The RMCM bit of MCR1 determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register(s) (RBR(s)). The receiver does not copy the content of the RBR(s) to the DRR(s), and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated and, if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1. Accepts bits shifted in from the DR pin in channel 0
2. Ignores bits received in channels 1-14
3. Accepts bits shifted in from the DR pin in channel 15
4. Ignores bits received in channels 16-38
5. Accepts bits shifted in from the DR pin in channel 39

20.6.7 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in [Section 20.6.7.1](#). The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in the following table.

Table 20-13. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
11b	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:...

1. Shifts data to the DX pin in channel 0
2. Places the DX pin in the high impedance state in channels 1-14
3. Shifts data to the DX pin in channel 15
4. Places the DX pin in the high impedance state in channels 16-38
5. Shifts data to the DX pin in channel 39

20.6.7.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed)
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled channel	A channel that can begin transmission by passing data from the data transmit register(s) (DXR(s)) to the transmit shift registers (XSR(s)).
Masked channel	A channel that cannot complete transmission. The DX pin is held in the high impedance state; data cannot be shifted out on the DX pin. In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.
Disabled channel	A channel that is not enabled. A disabled channel is also masked. Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated. The XEMPTY bit of SPCR2 is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR(s) is shifted out on the DX pin.

20.6.7.2 Activity on McBSP Pins for Different Values of XMCM

Figure 20-35 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLN1 = 0000011b: 4 words per frame

- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLLEN1, and XWDLEN1, respectively.

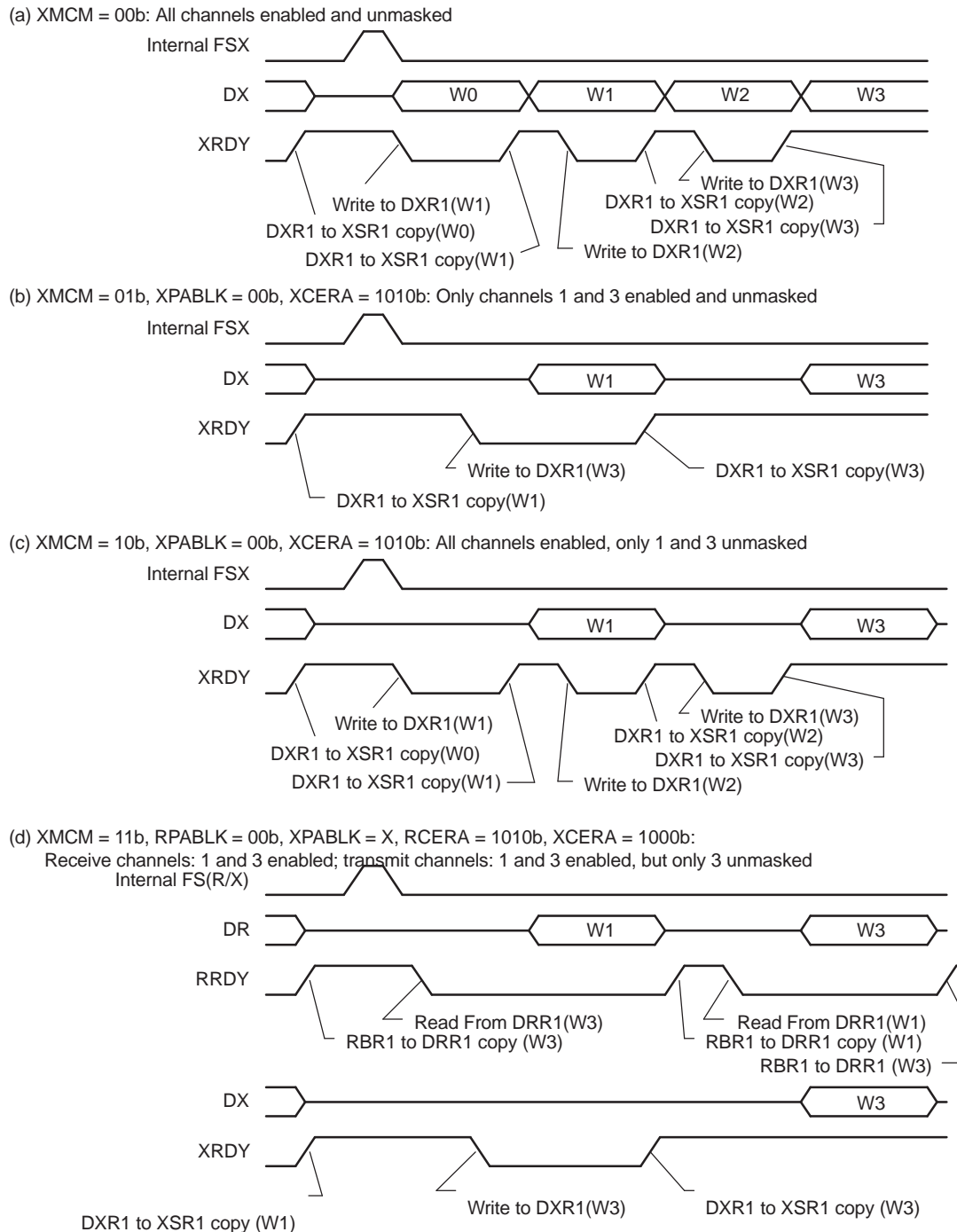
In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

20.6.8 Using Interrupts Between Block Transfers

When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 01b, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two CPU clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode (described in [Section 20.6.4](#)) and you want to know when you can assign a different block of channels to partition A or B.

Figure 20-35. Activity on McBSP Pins for the Possible Values of XMCM


20.7 SPI Operation Using the Clock Stop Mode

This chapter explains how to use the McBSP in SPI mode.

20.7.1 SPI Protocol

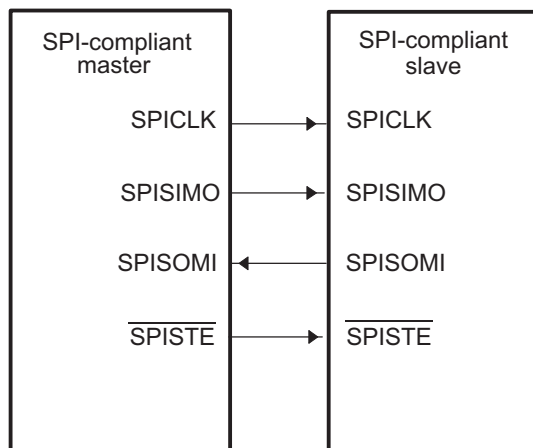
The SPI protocol is a master-slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

- Serial data input (also referred to as master in/slave out, or SPISOMI)

- Serial data output (also referred to as master out/slave in, or SPISIMO)
- Shift-clock (also referred to as SPICLK)
- Slave-enable signal (also referred to as $\overline{\text{SPISTE}}$)

A typical SPI interface with a single slave device is shown in [Figure 20-36](#).

Figure 20-36. Typical SPI Interface



The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. When the McBSP is operating in SPI master mode and the $\overline{\text{SPISTE}}$ signal is not used by the slave SPI port, the slave device must remain enabled at all times, and multiple slaves cannot be used.

20.7.2 Clock Stop Mode

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the serial clock signal (SPICLK) of the SPI protocol, while the transmit frame-synchronization signal (FSX) is used as the slave-enable signal ($\overline{\text{SPISTE}}$).

The receive clock signal (MCLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

20.7.3 Bits Used to Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in [Table 20-14](#). [Table 20-15](#) shows how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock stop mode configurations. The timing diagrams in [Section 20.7.4](#) show the effects of CLKSTP, CLKXP, and CLKRP.

Table 20-14. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP bits of SPCR1	Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 20-15 .)
CLKXP bit of PCR	This bit determines the polarity of the CLKX signal. (See also Table 20-15 .)
CLKRP bit of PCR	This bit determines the polarity of the MCLKR signal. (See also Table 20-15 .)
CLKXM bit of PCR	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).

Table 20-14. Bits Used to Enable and Configure the Clock Stop Mode (continued)

Bit Field	Description
XPHASE bit of XCR2	You must use a single-phase transmit frame (XPHASE = 0).
RPHASE bit of RCR2	You must use a single-phase receive frame (RPHASE = 0).
XFRLN1 bits of XCR1	You must use a transmit frame length of 1 serial word (XFRLN1 = 0).
RFRLN1 bits of RCR1	You must use a receive frame length of 1 serial word (RFRLN1 = 0).
XWDLEN1 bits of XCR1	The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.
RWDLEN1 bits of RCR1	The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode. The McBSP transmit and receive circuits are synchronized to a single clock.

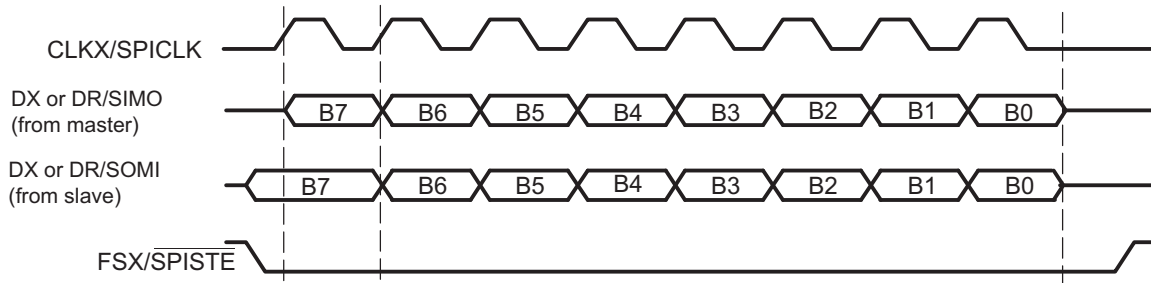
Table 20-15. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

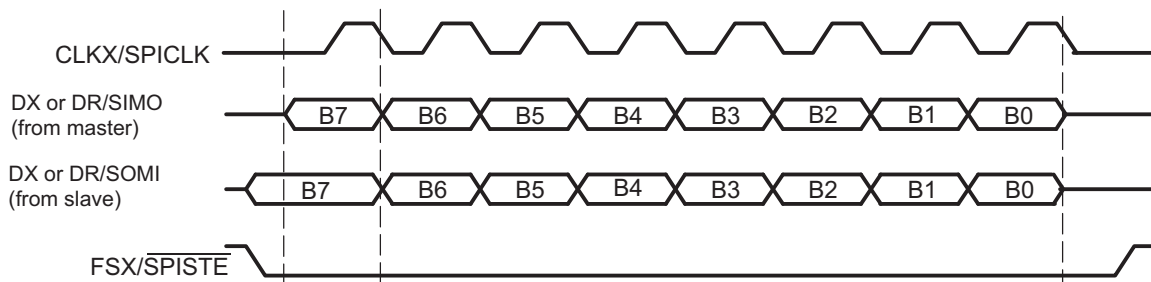
20.7.4 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown here. Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8, 12, 16, 20, 24, or 32 bits per packet. The receive packet length is selected with the RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

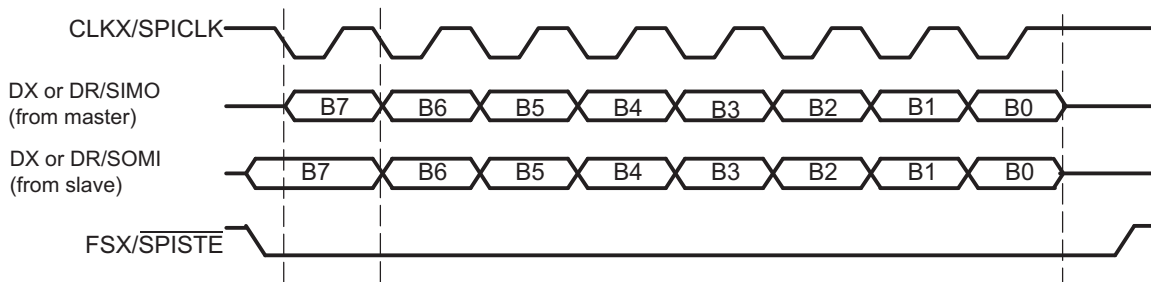
NOTE: Even if multiple words are consecutively transferred, the CLKX signal is always stopped and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer.

Figure 20-37. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 0, and CLKRP = 0


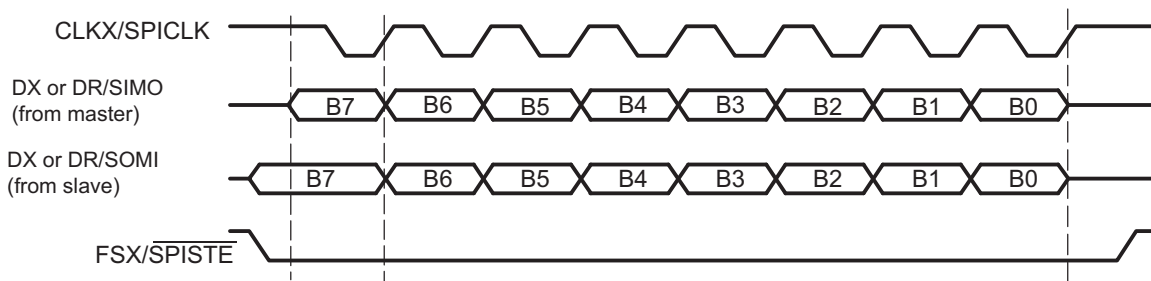
- A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
 B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

Figure 20-38. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 0, CLKRP = 1


- A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
 B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

Figure 20-39. SPI Transfer With CLKSTP = 10b (No Clock Delay), CLKXP = 1, and CLKRP = 0


- A If the McBSP is the SPI master (CLKXM = 1), SIMO = DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
 B If the McBSP is the SPI master (CLKXM = 1), SOMI = DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

Figure 20-40. SPI Transfer With CLKSTP = 11b (Clock Delay), CLKXP = 1, CLKRP = 1


- A If the McBSP is the SPI master (CLKXM = 1), SIMO=DX. If the McBSP is the SPI slave (CLKXM = 0), SIMO = DR.
 B If the McBSP is the SPI master (CLKXM = 1), SOMI=DR. If the McBSP is the SPI slave (CLKXM = 0), SOMI = DX.

20.7.5 Procedure for Configuring a McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

Step 1. Place the transmitter and receiver in reset.

Clear the transmitter reset bit (XRST = 0) in SPCR2 to reset the transmitter. Clear the receiver reset bit (RRST = 0) in SPCR1 to reset the receiver.

Step 2. Place the sample rate generator in reset.

Clear the sample rate generator reset bit (GRST = 0) in SPCR2 to reset the sample rate generator.

Step 3. Program registers that affect SPI operation.

Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bits settings, see one of the following topics:

- *McBSP as the SPI Master* ([Section 20.7.6](#))
- *McBSP as an SPI Slave* ([Section 20.7.7](#))

Step 4. Enable the sample rate generator.

To release the sample rate generator from reset, set the sample rate generator reset bit (GRST = 1) in SPCR2.

Make sure that during the write to SPCR2, you only modify GRST. Otherwise, you modify the McBSP configuration you selected in the previous step.

Step 5. Enable the transmitter and receiver.

After the sample rate generator is released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter (XRST = 1 in SPCR2) and enable the receiver (RRST = 1 in SPCR1).

If the DMA controller services the McBSP transmit and receive buffers, then you must first configure the DMA controller (this includes enabling the channels that service the McBSP buffers). When the DMA controller is ready, make XRST = 1 and RRST = 1.

In either case, make sure you only change XRST and RRST when you write to SPCR2 and SPCR1. Otherwise, you modify the bit settings you selected earlier in this procedure.

After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

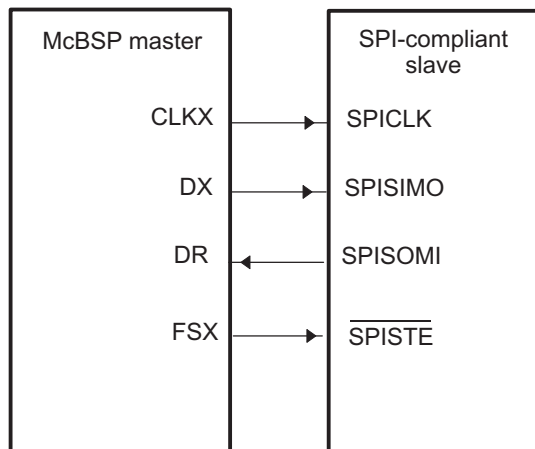
Step 6. If necessary, enable the frame-synchronization logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1,2] is loaded with data), set FRST = 1 if an internally generated frame-synchronization pulse is required (that is, if the McBSP is the SPI master).

20.7.6 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in [Figure 20-41](#). When the McBSP is configured as a master, the transmit output signal (DX) is used as the SPISIMO signal of the SPI protocol and the receive input signal (DR) is used as the SPISOMI signal.

The register bit values required to configure the McBSP as a master are listed in [Table 20-16](#). After the table are more details about the configuration requirements.

Figure 20-41. SPI Interface with McBSP Used as Master

Table 20-16. Bit Values Required to Configure the McBSP as an SPI Master

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The MCLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, MCLKR is driven internally by CLKX.
SCLKME = 0	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock.
CLKSM = 1	
CLKGDV is a value from 1 to 255	CLKGDV defines the divide down value for CLKG.
FSXM = 1	The FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-synchronization pulse on the FSX pin every time data is transferred from DXR1 to XSR1.
FSXP = 1	The FSX pin is active low.
XDATDLY = 01b	This setting provides the correct setup time on the FSX signal.
RDATDLY = 01b	

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the MCLKX pin is enabled only during packet transfers. When packets are not being transferred, the MCLKX pin remains high or low depending on the polarity used.

For SPI master operation, the MCLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the MCLKX pin to the MCLKR signal so that no external signal connection is required on the MCLKR pin and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

The McBSP can also provide a slave-enable signal (SS_) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output and the transmitter must be configured so that a frame-synchronization pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the FSX pin is programmable high or low; however, in most cases the pin must be configured active low.

When the McBSP is configured as described for SPI-master operation, the bit fields for frame-synchronization pulse width (FWID) and frame-synchronization period (FPER) are overridden, and custom frame-synchronization waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in [Section 20.7.4](#). The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

20.7.7 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in [Figure 20-42](#). When the McBSP is configured as a slave, DX is used as the SPISOMI signal and DR is used as the SPISIMO signal.

The register bit values required to configure the McBSP as a slave are listed in [Table 20-17](#). Following the table are more details about configuration requirements.

Figure 20-42. SPI Interface With McBSP Used as Slave

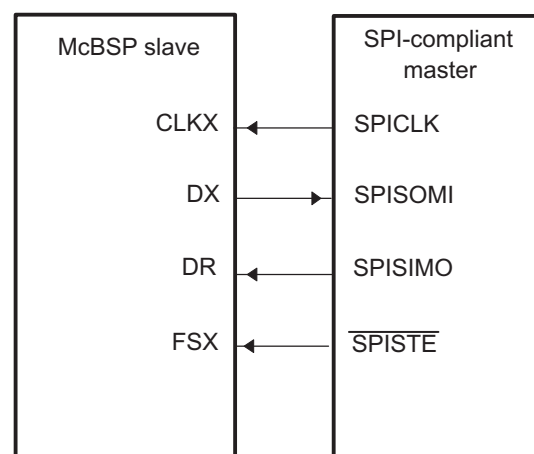


Table 20-17. Bit Values Required to Configure the McBSP as an SPI Slave

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the MCLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of MCLKR as seen on the MCLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The MCLKX pin is an input pin, so that it can be driven by the SPI master. Because CLKSTP = 10b or 11b, MCLKR is driven internally by CLKX.
SCLKME = 0	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.)
CLKSM = 1	
CLKGDV = 1	The sample rate generator divides the CPU clock before generating CLKG.
FSXM = 0	The FSX pin is an input pin, so that it can be driven by the SPI master.
FSXP = 1	The FSX pin is active low.
XDATDLY = 00b	These bits must be 0s for SPI slave operation.
RDATDLY = 00b	

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the CLKX and FSX pins must be configured as inputs. The MCLKX pin is internally connected to the MCLKR signal, so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the MCLKR and FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator must be programmed to its maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers. Unlike the standard SPI, this pin cannot be tied low all the time.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

20.8 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1. Place the McBSP/receiver in reset (see [Section 20.8.2](#)).
2. Program McBSP registers for the desired receiver operation (see [Section 20.8.1](#)).
3. Take the receiver out of reset (see [Section 20.8.2](#)).

20.8.1 Programming the McBSP Registers for the Desired Receiver Operation

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
 - Set the receiver pins to operate as McBSP pins.
 - Enable/disable the digital loopback mode.
 - Enable/disable the clock stop mode.
 - Enable/disable the receive multichannel selection mode.
- Data behavior:
 - Choose 1 or 2 phases for the receive frame.
 - Set the receive word length(s).
 - Set the receive frame length.
 - Enable/disable the receive frame-synchronization ignore function.
 - Set the receive companding mode.
 - Set the receive data delay.
 - Set the receive sign-extension and justification mode.
 - Set the receive interrupt mode.
- Frame-synchronization behavior:
 - Set the receive frame-synchronization mode.
 - Set the receive frame-synchronization polarity.
 - Set the sample rate generator (SRG) frame-synchronization period and pulse width.
- Clock behavior:
 - Set the receive clock mode.
 - Set the receive clock polarity.
 - Set the SRG clock divide-down value.
 - Set the SRG clock synchronization mode.
 - Set the SRG clock mode (choose an input clock).
 - Set the SRG input clock polarity.

20.8.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset). [Table 20-18](#) describes the bits used for both of these steps.

Table 20-18. Register Bits Used to Reset or Enable the McBSP Receiver Field Descriptions

Register	Bit	Field	Value	Description
SPCR2	7	FRST	0	Frame-synchronization logic reset Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1.
			1	If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.
SPCR2	6	GRST	0	Sample rate generator reset Sample rate generator is reset. If GRST = 0 due to a DSP reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).
			1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.
SPCR1	0	RRST	0	Receiver reset The serial port receiver is disabled and in the reset state.
			1	The serial port receiver is enabled.

20.8.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. The DSP reset (\overline{XRS} signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed (\overline{XRS} signal released), GRST = FRST = RRST = XRST = 0 keep the entire serial port in the reset state, provided the McBSP clock is turned on.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.

[Table 20-19](#) shows the state of McBSP pins when the serial port is reset due to a device reset and a direct receiver/transmitter reset.

For more details about McBSP reset conditions and effects, see [Section 20.10.2](#).

Table 20-19. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By Device Reset	State Forced By Receiver Reset (RRST = 0 and GRST = 1)
MDRx	I	GPIO Input	Input
MCLKRx	I/O/Z	GPIO Input	Known state if input; MCLKR running if output
MFSRx	I/O/Z	GPIO Input	Known state if input; FSRP inactive state if output Transmitter reset (XRST = 0 and GRST = 1)
MDXx	O/Z	GPIO Input	Low impedance after transmit bit clock provided
MCLKXx	I/O/Z	GPIO Input	Known state if input; CLKX running if output
MFSXx	I/O/Z	GPIO Input	Known state if input; FSXP inactive state if output

20.8.3 Set the Receiver Pins to Operate as McBSP Pins

To configure a pin for its McBSP function, you should configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.

20.8.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in [Table 20-20](#).

Table 20-20. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	15	DLB	Digital loopback mode	R/W	0
			DLB = 0		Digital loopback mode is disabled.
			DLB = 1		Digital loopback mode is enabled.

20.8.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in [Table 20-21](#). This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 20-21. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
MDR (receive data)	MDX (transmit data)
MFSR (receive frame synchronization)	MFSX (transmit frame synchronization)
MCLKR (receive clock)	MCLKX (transmit clock)

20.8.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in [Table 20-22](#).

Table 20-22. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode	R/W	00
			CLKSTP = 0Xb		Clock stop mode disabled; normal clocking for non-SPI mode
			CLKSTP = 10b		Clock stop mode enabled, without clock delay
			CLKSTP = 11b		Clock stop mode enabled, with clock delay

20.8.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

[Table 20-23](#) summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 20-23. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

20.8.6 Enable/Disable the Receive Multichannel Selection Mode

The RCMC bit determines whether the receive multichannel selection mode is on. RCMC is described in [Table 20-24](#). For more details, see [Section 20.6.6](#).

Table 20-24. Register Bit Used to Enable/Disable the Receive Multichannel Selection Mode

Register	Bit	Name	Function	Type	Reset Value
MCR1	0	RCMC	Receive multichannel selection mode RCMC = 0 The mode is disabled. All 128 channels are enabled. RCMC = 1 The mode is enabled. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.	R/W	0

20.8.7 Choose One or Two Phases for the Receive Frame

The RPHASE bit (see [Table 20-25](#)) determines whether the receive data frame has one or two phases.

Table 20-25. Register Bit Used to Choose One or Two Phases for the Receive Frame

Register	Bit	Name	Function	Type	Reset Value
RCR2	15	RPHASE	Receive phase number Specifies whether the receive frame has 1 or 2 phases. RPHASE = 0 Single-phase frame RPHASE = 1 Dual-phase frame	R/W	0

20.8.8 Set the Receive Word Length(s)

The RWDLEN1 and RWDLEN2 bit fields (see [Table 20-26](#)) determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

Table 20-26. Register Bits Used to Set the Receive Word Length(s)

Register	Bit	Name	Function	Type	Reset Value
RCR1	7-5	RWDLEN1	Receive word length 1	R/W	000
			Specifies the length of every serial word in phase 1 of the receive frame.		
			RWDLEN1 = 000		
			8 bits		
			RWDLEN1 = 001		
			12 bits		
			RWDLEN1 = 010		
			16 bits		
			RWDLEN1 = 011		
RCR2	7-5	RWDLEN2	Receive word length 2	R/W	000
			If a dual-phase frame is selected, RWDLEN2 specifies the length of every serial word in phase 2 of the frame.		
			RWDLEN2 = 000		
			8 bits		
			RWDLEN2 = 001		
			12 bits		
			RWDLEN2 = 010		
			16 bits		
			RWDLEN2 = 011		
			20 bits		
			RWDLEN2 = 100		
			24 bits		
			RWDLEN2 = 101		
			32 bits		
			RWDLEN2 = 11X		
			Reserved		

20.8.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame and RWDLEN2 determines the word length in phase 2 of the frame.

20.8.9 Set the Receive Frame Length

The RFRLN1 and RFRLN2 bit fields (see [Table 20-27](#)) determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

Table 20-27. Register Bits Used to Set the Receive Frame Length

Register	Bit	Name	Function	Type	Reset Value
RCR1	14-8	RFRLN1	Receive frame length 1	R/W	000 0000
			(RFRLN1 + 1) is the number of serial words in phase 1 of the receive frame.		
			RFRLN1 = 000 0000		
			1 word in phase 1		
			RFRLN1 = 000 0001		
			2 words in phase 1		
			RFRLN1 = 111 1111		
			128 words in phase 1		

Table 20-27. Register Bits Used to Set the Receive Frame Length (continued)

Register	Bit	Name	Function	Type	Reset Value
RCR2	14-8	RFLEN2	Receive frame length 2 If a dual-phase frame is selected, (RFLEN2 + 1) is the number of serial words in phase 2 of the receive frame. RFLEN2 = 000 0000 1 word in phase 2 RFLEN2 = 000 0001 2 words in phase 2 RFLEN2 = 111 1111 128 words in phase 2	R/W	000 0000

20.8.9.1 Selected Frame Length

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit RFLEN fields allow up to 128 words per phase. See [Table 20-28](#) for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFLEN1.

Table 20-28. How to Calculate the Length of the Receive Frame

RPHASE	RFLEN1	RFLEN2	Frame Length
0	$0 \leq \text{RFLEN1} \leq 127$	Don't care	(RFLEN1 + 1) words
1	$0 \leq \text{RFLEN1} \leq 127$	$0 \leq \text{RFLEN2} \leq 127$	(RFLEN1 + 1) + (RFLEN2 + 1) words

20.8.10 Enable/Disable the Receive Frame-Synchronization Ignore Function

The RFIG bit (see [Table 20-29](#)) controls the receive frame-synchronization ignore function.

Table 20-29. Register Bit Used to Enable/Disable the Receive Frame-Synchronization Ignore Function

Register	Bit	Name	Function	Type	Reset Value
RCR2	2	RFIG	Receive frame-synchronization ignore RFIG = 0 An unexpected receive frame-synchronization pulse causes the McBSP to restart the frame transfer. RFIG = 1 The McBSP ignores unexpected receive frame-synchronization pulses.	R/W	0

20.8.10.1 Unexpected Frame-Synchronization Pulses and the Frame-Synchronization Ignore Function

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-synchronization pulses.

When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

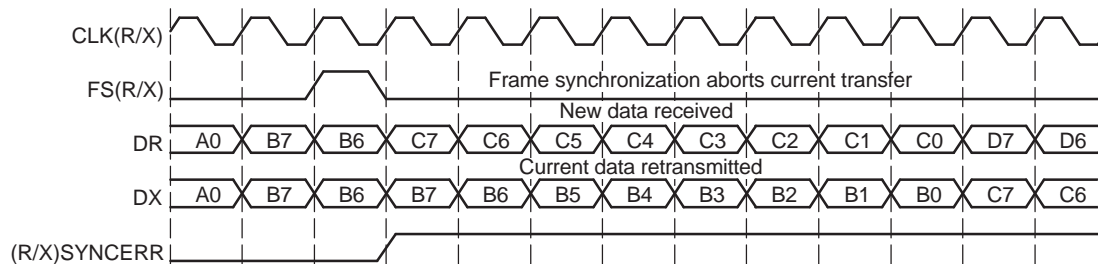
1. Aborts the current data transfer
2. Sets RSYNCERR in SPCR1 to 1
3. Begins the transfer of a new data word

For more details about the frame-synchronization error condition, see [Section 20.5.3](#).

20.8.10.2 Examples of Effects of RFIG

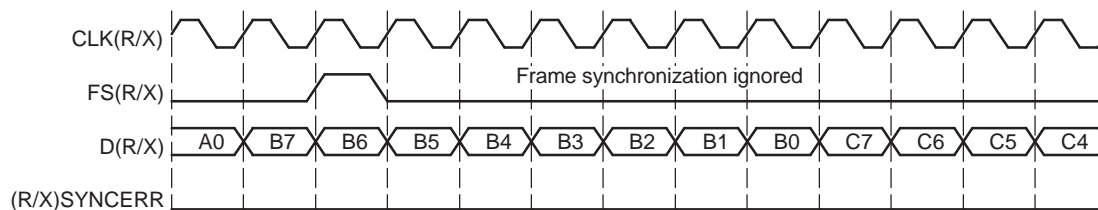
[Figure 20-43](#) shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word © in this example) is received after the appropriate data delay. This condition is a receive synchronization error, which sets the RSYNCERR bit.

Figure 20-43. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 0



In contrast with [Figure 20-43](#), [Figure 20-44](#) shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 20-44. Unexpected Frame-Synchronization Pulse With (R/X)FIG = 1



20.8.11 Set the Receive Companding Mode

The RCOMPAND bits (see [Table 20-30](#)) determine whether companding or another data transfer option is chosen for McBSP reception.

Table 20-30. Register Bits Used to Set the Receive Companding Mode

Register	Bit	Name	Function	Type	Reset Value
RRCR2	4-3	RCOMPAND	Receive companding mode	R/W	00
Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data.					
RCOMPAND = 00 No companding, any size data, MSB received first					
RCOMPAND = 01 No companding, 8-bit data, LSB received first (for details, see Section 20.8.11.4).					
RCOMPAND = 10 μ -law companding, 8-bit data, MSB received first					
RCOMPAND = 11 A-law companding, 8-bit data, MSB received first					

20.8.11.1 Companding

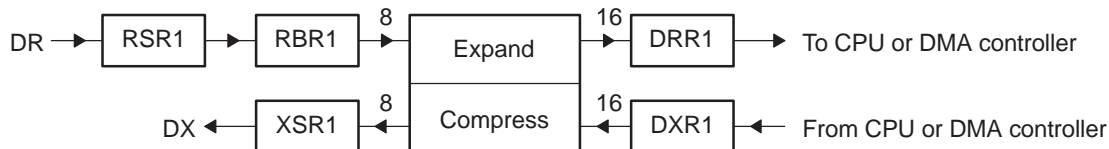
Companding (COMpressing and exPANDING) hardware allows compression and expansion of data in either μ -law or A-law format. The companding standard employed in the United States and Japan is μ -law. The European companding standard is referred to as A-law. The specifications for μ -law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ -law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

The μ -law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 20-45 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ -law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2's-complement format.

Figure 20-45. Companding Processes for Reception and for Transmission



20.8.11.2 Format of Expanded Data

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The RJUST bit of SPCR1 is ignored when companding is used.

20.8.11.3 Companding Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See [Section 20.3.2.2](#).

20.8.11.4 Option to Receive LSB First

Normally, the McBSP transmits or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.

20.8.12 Set the Receive Data Delay

The RDATDLY bits (see [Table 20-31](#)) determine the length of the data delay for the receive frame.

Table 20-31. Register Bits Used to Set the Receive Data Delay

Register	Bit	Name	Function	Type	Reset Value
RCR2	1-0	RDATDLY	Receive data delay	R/W	00
			RDATDLY = 00		0-bit data delay
			RDATDLY = 01		1-bit data delay
			RDATDLY = 10		2-bit data delay
			RDATDLY = 11		Reserved

20.8.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

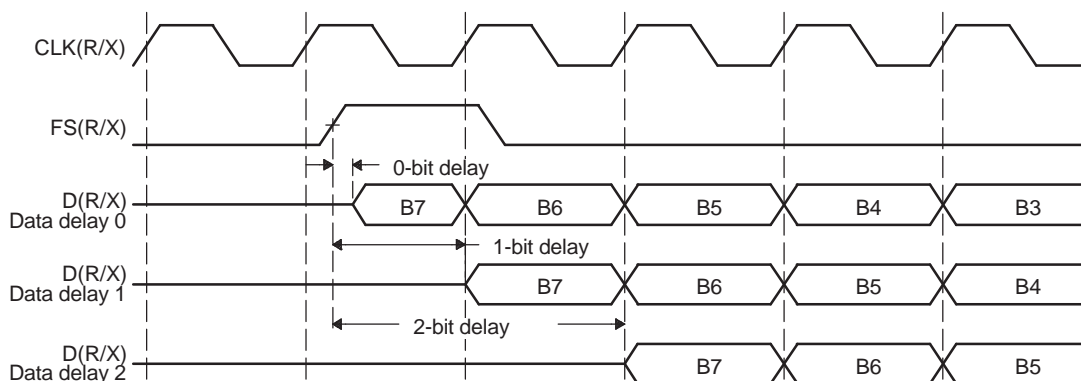
RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit-clocks (RDATDLY = 00b-10b), as described in Table 20-31 and shown in Figure 20-46. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

20.8.12.2 0-Bit Data Delay

Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

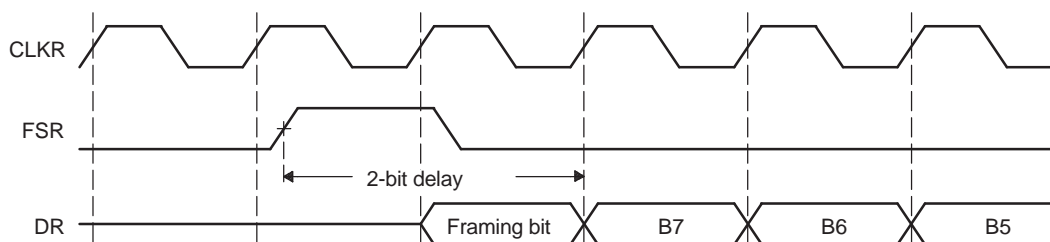
Figure 20-46. Range of Programmable Data Delay



20.8.12.3 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 20-47. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 20-47. 2-Bit Data Delay Used to Skip a Framing Bit



20.8.13 Set the Receive Sign-Extension and Justification Mode

The RJUST bits (see [Table 20-32](#)) determine whether data received by the McBSP is sign-extended and how it is justified.

Table 20-32. Register Bits Used to Set the Receive Sign-Extension and Justification Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	14-13	RJUST	Receive sign-extension and justification mode	R/W	00
			RJUST = 00 Right justify data and zero fill MSBs in DRR[1,2]		
			RJUST = 01 Right justify data and sign extend it into the MSBs in DRR[1,2]		
			RJUST = 10 Left justify data and zero fill LSBs in DRR[1,2]		
			RJUST = 11 Reserved		

20.8.13.1 Sign-Extension and the Justification

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and whether unused bits in DRR[1,2] are filled with zeros or with sign bits.

[Table 20-33](#) and [Table 20-34](#) show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value ABCh. The second table shows the effect on an example 20-bit receive-data value ABCDEh.

Table 20-33. Example: Use of RJUST Field With 12-Bit Data Value ABCh

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0000h	0ABCh
01b	Right	Sign extend data into MSBs	FFFFh	FABCh
10b	Left	Zero fill LSBs	0000h	ABC0h
11b	Reserved	Reserved	Reserved	Reserved

Table 20-34. Example: Use of RJUST Field With 20-Bit Data Value ABCDEh

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	000Ah	BCDEh
01b	Right	Sign extend data into MSBs	FFFAh	BCDEh
10b	Left	Zero fill LSBs	ABCDh	E000h
11b	Reserved	Reserved	Reserved	Reserved

20.8.14 Set the Receive Interrupt Mode

The RINTM bits (see [Table 20-35](#)) determine which event generates a receive interrupt request to the CPU.

The receive interrupt (RINT) informs the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

Table 20-35. Register Bits Used to Set the Receive Interrupt Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	5-4	RINTM	Receive interrupt mode	R/W	00
			RINTM = 00 RINT generated when RRDY changes from 0 to 1. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.		
			RINTM = 01 RINT generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode. In the multichannel selection mode, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 20.6.8 . In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			RINTM = 10 RINT generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via RINT.		
			RINTM = 11 RINT generated when RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 20.5.3 .		

20.8.15 Set the Receive Frame-Synchronization Mode

The bits described in [Table 20-36](#) determine the source for receive frame synchronization and the function of the FSR pin.

20.8.15.1 Receive Frame-Synchronization Modes

[Table 20-37](#) shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

In digital loopback mode (DLB = 1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 20-36. Register Bits Used to Set the Receive Frame Synchronization Mode

Register	Bit	Name	Function	Type	Reset Value
PCR	10	FSRM	Receive frame-synchronization mode	R/W	0
			FSRM = 0 Receive frame synchronization is supplied by an external source via the FSR pin.		
			FSRM = 1 Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.		

Table 20-36. Register Bits Used to Set the Receive Frame Synchronization Mode (continued)

Register	Bit	Name	Function	Type	Reset Value
SRGR2	15	GSYNC	<p>Sample rate generator clock synchronization mode</p> <p>If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin.</p> <p>GSYNC = 0 No clock synchronization is used: CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.</p> <p>GSYNC = 1 Clock synchronization is used. When a pulse is detected on the FSR pin:</p> <ul style="list-style-type: none"> • CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR pin. • FSG pulses FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. <p>For more details, see Section 20.4.3.</p>	R/W	0
SPCR1	15	DLB	<p>Digital loopback mode</p> <p>DLB = 0 Digital loopback mode is disabled.</p> <p>DLB = 1 Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.</p>	R/W	0
SPCR1	12-11	CLKSTP	<p>Clock stop mode</p> <p>CLKSTP = 0xb Clock stop mode disabled; normal clocking for non-SPI mode.</p> <p>CLKSTP = 10b Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.</p> <p>CLKSTP = 11b Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.</p>	R/W	00

Table 20-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
0	0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Internal FSR is driven by the sample rate generator frame-synchronization signal (FSG).	Input. The external frame-synchronization input on the FSR pin is used to synchronize CLKG and generate FSG pulses.
1	0	0	Internal FSX drives internal FSR.	High impedance

Table 20-37. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin (continued)

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
1	0 or 1	1	Internal FSX drives internal FSR.	Input. If the sample rate generator is running, external FSR is used to synchronize CLKG and generate FSG pulses.
1	1	0	Internal FSX drives internal FSR.	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out on the FSR pin.

20.8.16 Set the Receive Frame-Synchronization Polarity

The FSRP bit (see [Table 20-38](#)) determines whether frame-synchronization pulses are active high or active low on the FSR pin.

Table 20-38. Register Bit Used to Set Receive Frame-Synchronization Polarity

Register	Bit	Name	Function	Type	Reset Value
PCR	2	FSRP	Receive frame-synchronization polarity	R/W	0
			FSRP = 0		Frame-synchronization pulse FSR is active high.
			FSRP = 1		Frame-synchronization pulse FSR is active low.

20.8.16.1 Frame-Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see [Section 20.4.2](#)) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see [Section 20.8.15](#). Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see [Section 20.8.17](#)).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

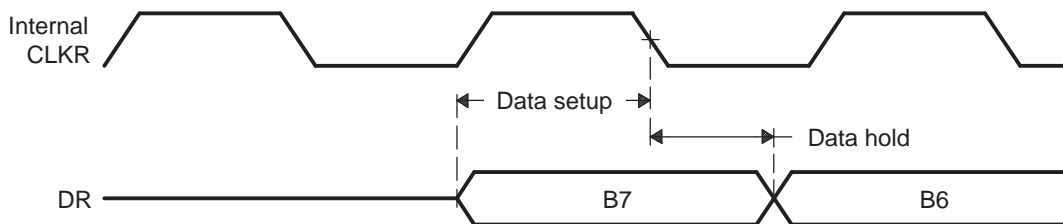
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

MCLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. [Figure 20-48](#) shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 20-48. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



Set the SRG Frame-Synchronization Period and Pulse Width

20.8.16.2 Frame-Synchronization Period and the Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0. [Table 20-39](#) shows settings for FPER and FWID.

[Figure 20-49](#) shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

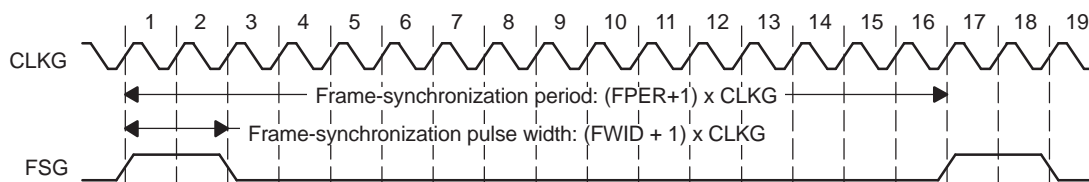
Table 20-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset Value
SRGR2	11-0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles	R/W	0000 0000 0000
SRGR1	15-8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG.	R/W	0000 0000

Table 20-39. Register Bits Used to Set the SRG Frame-Synchronization Period and Pulse Width (continued)

Register	Bit	Name	Function	Type	Reset Value
Range for (FWID + 1): 1 to 256 CLKG cycles					

Figure 20-49. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

20.8.17 Set the Receive Clock Mode

Table 20-40 shows the settings for bits used to set receive clock mode.

Table 20-40. Register Bits Used to Set the Receive Clock Mode

Register	Bit	Name	Function	Type	Reset Value
PCR	8	CLKRM	Receive clock mode	R/W	0
			Case 1: Digital loopback mode not set (DLB = 0) in SPCR1.		
			CLKRM = 0 The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR).		
			CLKRM = 1 Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR.		
SPCR1	15	DLB	Case 2: Digital loopback mode set (DLB = 1) in SPCR1.	R/W	00
			CLKRM = 0 The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). Internal CLKX is derived according to the CLKXM bit of PCR.		
			CLKRM = 1 Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. Internal CLKX is derived according to the CLKXM bit of PCR.		
			DLB = 0 Digital loopback mode is disabled.		
			DLB = 1 Digital loopback mode is enabled. The receive signals, including the receive frame-synchronization signal, are connected internally through multiplexers to the corresponding transmit signals.		

Table 20-40. Register Bits Used to Set the Receive Clock Mode (continued)

Register	Bit	Name	Function	Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode	R/W	00
			CLKSTP = 0xb		Clock stop mode disabled; normal clocking for non-SPI mode.
			CLKSTP = 10b		Clock stop mode enabled without clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.
			CLKSTP = 11b		Clock stop mode enabled with clock delay. The internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

20.8.17.1 Selecting a Source for the Receive Clock and a Data Direction for the MCLKR Pin

Table 20-41 shows how you can select various sources to provide the receive clock signal and affect the MCLKR pin. The polarity of the signal on the MCLKR pin is determined by the CLKRP bit.

In the digital loopback mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in the clock stop mode, the internal receive clock signal (MCLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 20-41. Receive Clock Signal Source Selection

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	MCLKR Pin Status
0	0	The MCLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
0	1	The sample rate generator clock (CLKG) drives internal MCLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the MCLKR pin.
1	0	Internal CLKX drives internal MCLKR. To configure CLKX, see Section 20.9.18 .	High impedance
1	1	Internal CLKX drives internal MCLKR. To configure CLKX, see Section 20.9.18 .	Output. Internal MCLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the MCLKR pin.

20.8.18 Set the Receive Clock Polarity

Table 20-42. Register Bit Used to Set Receive Clock Polarity

Register	Bit	Name	Function	Type	Reset Value
PCR	0	CLKRP	Receive clock polarity	R/W	0
			CLKRP = 0		Receive data sampled on falling edge of MCLKR
			CLKRP = 1		Receive data sampled on rising edge of MCLKR

20.8.18.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Receive frame-synchronization pulses can be generated internally by the sample rate generator (see [Section 20.4.2](#)) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see [Section 20.8.15](#). Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR (see [Section 20.8.17](#)).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

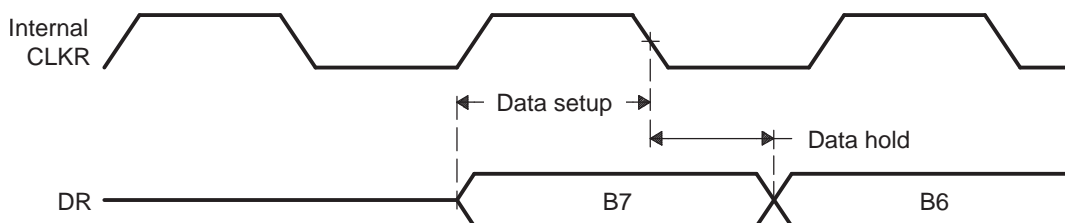
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. [Figure 20-50](#) shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 20-50. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



20.8.19 Set the SRG Clock Divide-Down Value

Table 20-43. Register Bits Used to Set the Sample Rate Generator (SRG) Clock Divide-Down Value

Register	Bit	Name	Function	Type	Reset Value
SRGR1	7-0	CLKGDV	Sample rate generator clock divide-down value The input clock of the sample rate generator is divided by (CLKGDV + 1) to generate the required sample rate generator clock frequency. The default value of CLKGDV is 1 (divide input clock by 2).	R/W	0000 0001

20.8.19.1 Sample Rate Generator Clock Divider

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to $1/(\text{CLKGDV} + 1)$ of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p + 1 cycles and the low-state duration is p cycles.

20.8.20 Set the SRG Clock Synchronization Mode

For more details on using the clock synchronization feature, see [Section 20.4.3](#).

Table 20-44. Register Bit Used to Set the SRG Clock Synchronization Mode

Register	Bit	Name	Function	Type	Reset Value
SRGR2	15	GSYNC	Sample rate generator clock synchronization GSYNC is used only when the input clock source for the sample rate generator is external—on the MCLKR or MCLKX pin. GSYNC = 0 The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles. GSYNC = 1 Clock synchronization is performed. When a pulse is detected on the FSR pin: <ul style="list-style-type: none"> • CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR or MCLKX pin. • FSG pulses. FSG only pulses in response to a pulse on the FSR pin. The frame-synchronization period defined in FPER is ignored. 	R/W	0

20.8.21 Set the SRG Clock Mode (Choose an Input Clock)

Table 20-45. Register Bits Used to Set the SRG Clock Mode (Choose an Input Clock)

Register	Bit	Name	Function	Type	Reset Value
PCR	7	SCLKME	Sample rate generator clock mode	R/W	0
SRGR2	13	CLKSM		R/W	1
			SCLKME = 0 CLKSM = 0	Reserved	
			SCLKME = 0 CLKSM = 1	Sample rate generator clock derived from LSPCLK (default)	
			SCLKME = 1 CLKSM = 0	Sample rate generator clock derived from MCLKR pin	
			SCLKME = 1 CLKSM = 1	Sample rate generator clock derived from MCLKX pin	

20.8.21.1 SRG Clock Mode

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. [Table 20-45](#) shows the four possible sources of the input clock. For more details on generating CLKG, see [Section 20.4.1.1](#).

20.8.22 Set the SRG Input Clock Polarity

Table 20-46. Register Bits Used to Set the SRG Input Clock Polarity

Register	Bit	Name	Function	Type	Reset Value
PCR	1	CLKXP	MCLKX pin polarity CLKXP determines the input clock polarity when the MCLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1). CLKXP = 0 Rising edge on MCLKX pin generates transitions on CLKG and FSG. CLKXP = 1 Falling edge on MCLKX pin generates transitions on CLKG and FSG.	R/W	0
PCR	0	CLKRP	MCLKR pin polarity CLKRP determines the input clock polarity when the MCLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0). CLKRP = 0 Falling edge on MCLKR pin generates transitions on CLKG and FSG. CLKRP = 1 Rising edge on MCLKR pin generates transitions on CLKG and FSG.	R/W	0

20.8.22.1 Using CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKX or MCLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKXP for the MCLKX pin, CLKRP for the MCLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

20.9 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1. Place the McBSP/transmitter in reset (see [Section 20.9.2](#)).
2. Program the McBSP registers for the desired transmitter operation (see [Section 20.9.1](#)).
3. Take the transmitter out of reset (see [Section 20.9.2](#)).

20.9.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

- Global behavior:
 - Set the transmitter pins to operate as McBSP pins.
 - Enable/disable the digital loopback mode.
 - Enable/disable the clock stop mode.
 - Enable/disable transmit multichannel selection.
- Data behavior:
 - Choose 1 or 2 phases for the transmit frame.
 - Set the transmit word length(s).
 - Set the transmit frame length.
 - Enable/disable the transmit frame-synchronization ignore function.
 - Set the transmit companding mode.
 - Set the transmit data delay.

- Set the transmit DXENA mode.
- Set the transmit interrupt mode.
- Frame-synchronization behavior:
 - Set the transmit frame-synchronization mode.
 - Set the transmit frame-synchronization polarity.
 - Set the SRG frame-synchronization period and pulse width.
- Clock behavior:
 - Set the transmit clock mode.
 - Set the transmit clock polarity.
 - Set the SRG clock divide-down value.
 - Set the SRG clock synchronization mode.
 - Set the SRG clock mode (choose an input clock).
 - Set the SRG input clock polarity.

20.9.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset). [Table 20-47](#) describes the bits used for both of these steps.

Table 20-47. Register Bits Used to Place Transmitter in Reset Field Descriptions

Register	Bit	Field	Value	Description
SPCR2	7	FRST	0	Frame-synchronization logic reset Frame-synchronization logic is reset. The sample rate generator does not generate frame-synchronization signal FSG, even if GRST = 1.
			1	Frame-synchronization is enabled. If GRST = 1, frame-synchronization signal FSG is generated after (FPER + 1) number of CLKG clock cycles; all frame counters are loaded with their programmed values.
SPCR2	6	GRST	0	Sample rate generator reset Sample rate generator is reset. If GRST = 0 due to a device reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).
			1	Sample rate generator is enabled. CLKG is driven according to the configuration programmed in the sample rate generator registers (SRGR[1,2]). If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.
SPCR2	0	XRST	0	Transmitter reset The serial port transmitter is disabled and in the reset state.
			1	The serial port transmitter is enabled.

20.9.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1. A DSP reset ($\overline{\text{XRS}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed, GRST = FRST = RRST = XRST = 0, keeping the entire serial port in the reset state.
2. The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.
3. When using the DMA, the order in which McBSP events must occur is important. DMA channel and peripheral interrupts must be configured prior to releasing the McBSP transmitter from reset.

The reason for this is that an XRDY is fired when XRST = 1. The XRDY signals the DMA to start copying data from the buffer into the transmit register. If the McBSP transmitter is released from reset before the DMA channel and peripheral interrupts are configured, the XRDY signals before the DMA channel can receive the signal; therefore, the DMA does not move the data from the buffer to the

transmit register. The DMA PERINTFLG is edge-sensitive and will fail to recognize the XRDY, which is continuously high.

For more details about McBSP reset conditions and effects, see [Section 20.10.2](#).

20.9.3 Set the Transmitter Pins to Operate as McBSP Pins

To configure a pin for its McBSP function, you should configure the bits of the GPxMUXn register appropriately. In addition to this, bits 12 and 13 of the PCR register must be set to 0. These bits are defined as reserved.

20.9.4 Enable/Disable the Digital Loopback Mode

The DLB bit determines whether the digital loopback mode is on. DLB is described in [Table 20-48](#).

Table 20-48. Register Bit Used to Enable/Disable the Digital Loopback Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	15	DLB	Digital loopback mode	R/W	0
			DLB = 0		Digital loopback mode is disabled.
			DLB = 1		Digital loopback mode is enabled.

20.9.4.1 Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in [Table 20-49](#). This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 20-49. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This Receive Signal	Is Fed Internally by This Transmit Signal
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
MCLKR (receive clock)	CLKX (transmit clock)

20.9.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on. CLKSTP is described in [Table 20-50](#).

Table 20-50. Register Bits Used to Enable/Disable the Clock Stop Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	12-11	CLKSTP	Clock stop mode	R/W	00
			CLKSTP = 0xb		Clock stop mode disabled; normal clocking for non-SPI mode.
			CLKSTP = 10b		Clock stop mode enabled without clock delay
			CLKSTP = 11b		Clock stop mode enabled with clock delay

20.9.5.1 Clock Stop Mode

The clock stop mode supports the SPI master-slave protocol. If you do not plan to use the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the MCLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the MCLKR pin.

[Table 20-51](#) summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. In the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-synchronization signal is tied internally to the transmit frame-synchronization signal.

Table 20-51. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of MCLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of MCLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of MCLKR.

20.9.6 Enable/Disable Transmit Multichannel Selection

For more details, see [Section 20.6.7](#).

Table 20-52. Register Bits Used to Enable/Disable Transmit Multichannel Selection

Register	Bit	Name	Function	Type	Reset Value
MCR2	1-0	XMCM	Transmit multichannel selection	R/W	00
			XMCM = 00b No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.		
			XMCM = 01b All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked. The XMCM bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 10b All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCM bit determines whether 32 channels or 128 channels are selectable in XCERs.		
			XMCM = 11b This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCM bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.		

20.9.7 Choose One or Two Phases for the Transmit Frame

Table 20-53. Register Bit Used to Choose 1 or 2 Phases for the Transmit Frame

Register	Bit	Name	Function	Type	Reset Value
XCR2	15	XPHASE	Transmit phase number Specifies whether the transmit frame has 1 or 2 phases. XPHASE = 0 Single-phase frame XPHASE = 1 Dual-phase frame	R/W	0

20.9.8 Set the Transmit Word Length(s)

Table 20-54. Register Bits Used to Set the Transmit Word Length(s)

Register	Bit	Name	Function	Type	Reset Value
XCR1	7-5	XWDLEN1	Transmit word length of frame phase 1 XWDLEN1 = 000b 8 bits XWDLEN1 = 001b 12 bits XWDLEN1 = 010b 16 bits XWDLEN1 = 011b 20 bits XWDLEN1 = 100b 24 bits XWDLEN1 = 101b 32 bits XWDLEN1 = 11Xb Reserved	R/W	000
XCR2	7-5	XWDLEN2	Transmit word length of frame phase 2 XWDLEN2 = 000b 8 bits XWDLEN2 = 001b 12 bits XWDLEN2 = 010b 16 bits XWDLEN2 = 011b 20 bits XWDLEN2 = 100b 24 bits XWDLEN2 = 101b 32 bits XWDLEN2 = 11Xb Reserved	R/W	000

20.9.8.1 Word Length Bits

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, XWDLEN1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.

20.9.9 Set the Transmit Frame Length

Table 20-55. Register Bits Used to Set the Transmit Frame Length

Register	Bit	Name	Function	Type	Reset Value
XCR1	14-8	XFRLEN1	Transmit frame length 1	R/W	000 0000
			(XFRLEN1 + 1) is the number of serial words in phase 1 of the transmit frame.		
			XFRLEN1 = 000 0000		
			1 word in phase 1		
			XFRLEN1 = 000 0001		
XCR2	14-8	XFRLEN2	Transmit frame length 2	R/W	000 0000
			If a dual-phase frame is selected, (XFRLEN2 + 1) is the number of serial words in phase 2 of the transmit frame.		
			XFRLEN2 = 000 0000		
			1 word in phase 2		
			XFRLEN2 = 000 0001		

20.9.9.1 Selected Frame Length

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on the value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See [Table 20-56](#) for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

NOTE: Program the XFRLEN fields with $[w \text{ minus } 1]$, where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

Table 20-56. How to Calculate Frame Length

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	$0 \leq \text{XFRLEN1} \leq 127$	Don't care	(XFRLEN1 + 1) words
1	$0 \leq \text{XFRLEN1} \leq 127$	$0 \leq \text{XFRLEN2} \leq 127$	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

20.9.10 Enable/Disable the Transmit Frame-Synchronization Ignore Function

Table 20-57. Register Bit Used to Enable/Disable the Transmit Frame-Synchronization Ignore Function

Register	Bit	Name	Function	Type	Reset Value
XCR2	2	XFIG	Transmit frame-synchronization ignore	R/W	0
			XFIG = 0		An unexpected transmit frame-synchronization pulse causes the McBSP to restart the frame transfer.
			XFIG = 1		The McBSP ignores unexpected transmit frame-synchronization pulses.

20.9.10.1 Unexpected Frame-Synchronization Pulses and Frame-Synchronization Ignore

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse.

When XFIG = 1, normal transmission continues with unexpected frame-synchronization signals ignored.

When XFIG = 0 and an unexpected frame-synchronization pulse occurs, the serial port:

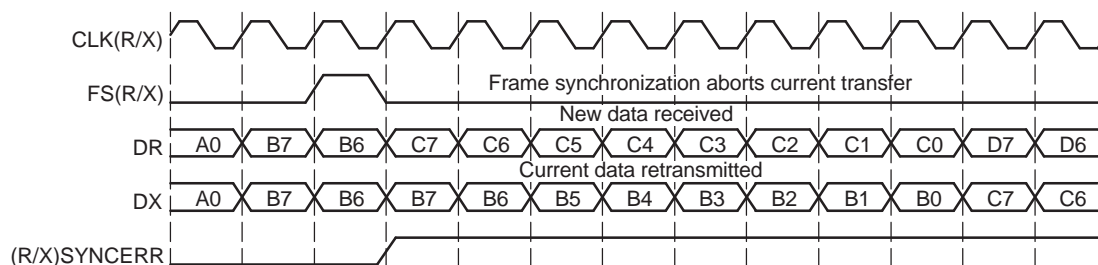
1. Aborts the present transmission
2. Sets XSYNCERR to 1 in SPCR2
3. Reinitiates transmission of the current word that was aborted

For more details about the frame-synchronization error condition, see [Section 20.5.6](#).

20.9.10.2 Examples Showing the Effects of XFIG

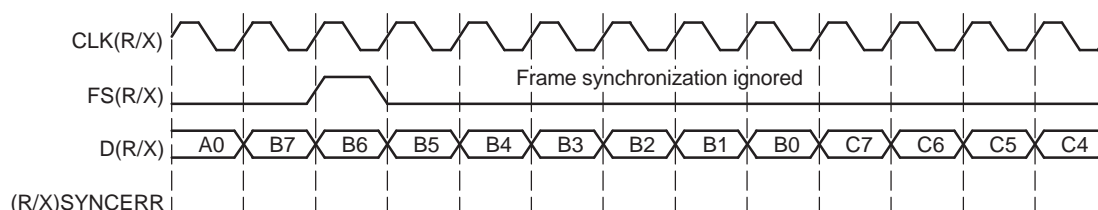
[Figure 20-51](#) shows an example in which word B is interrupted by an unexpected frame-synchronization pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, which sets the XSYNCERR bit. No new data has been written to DXR[1,2]; therefore, the McBSP transmits B again.

Figure 20-51. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 0



In contrast with [Figure 20-51](#), [Figure 20-52](#) shows McBSP operation when unexpected frame-synchronization signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-synchronization pulse.

Figure 20-52. Unexpected Frame-Synchronization Pulse With (R/X) FIG = 1



20.9.11 Set the Transmit Companding Mode

Table 20-58. Register Bits Used to Set the Transmit Companding Mode

Register	Bit	Name	Function	Type	Reset Value
XCR2	4-3	XCOMPAND	Transmit companding mode Modes other than 00b are enabled only when the appropriate XWDLEN is 000b, indicating 8-bit data. XCOMPAND = 00b No companding, any size data, MSB transmitted first XCOMPAND = 01b No companding, 8-bit data, LSB transmitted first (for details, see Section 20.8.11.4, Option to Receive LSB First) XCOMPAND = 10b μ -law companding, 8-bit data, MSB transmitted first XCOMPAND = 11b A-law companding, 8-bit data, MSB transmitted first	R/W	00

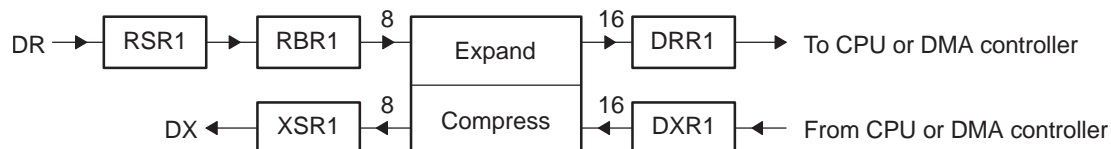
20.9.11.1 Companding

Companding (COMpressing and exPANDING) hardware allows compression and expansion of data in either μ -law or A-law format. The companding standard employed in the United States and Japan is μ -law. The European companding standard is referred to as A-law. The specifications for μ -law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ -law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or DMA controller must be at least 16 bits wide.

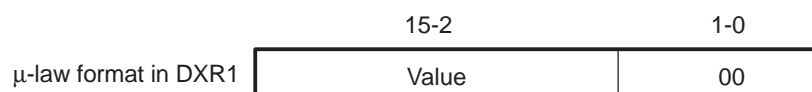
The μ -law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

[Figure 20-53](#) illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ -law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to two's-complement format.

Figure 20-53. Companding Processes for Reception and for Transmission


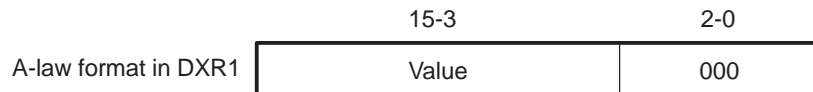
20.9.11.2 Format for Data To Be Compressed

For transmission using μ -law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in [Figure 20-54](#).

Figure 20-54. μ -Law Transmit Data Companding Format


For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in [Figure 20-55](#).

Figure 20-55. A-Law Transmit Data Companding Format



20.9.11.3 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See [Section 20.3.2.2, Capability to Compand Internal Data](#).

20.9.11.4 Option to Transmit LSB First

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits and LSB-first ordering is done.

20.9.12 Set the Transmit Data Delay

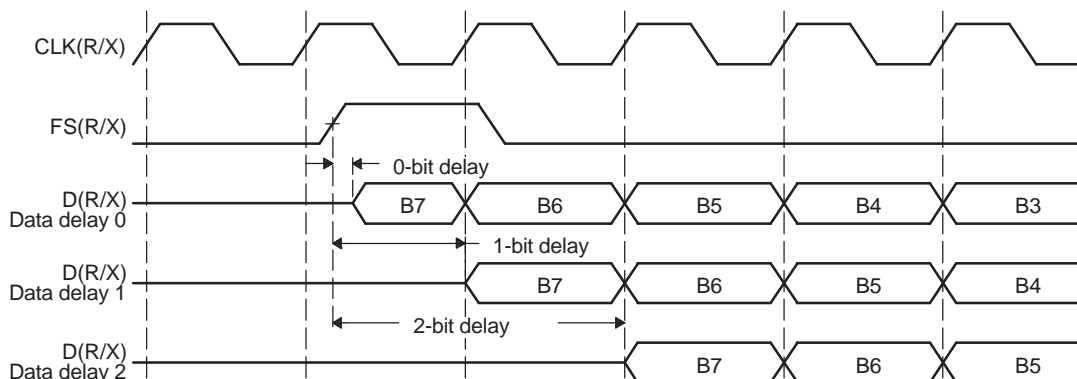
Table 20-59. Register Bits Used to Set the Transmit Data Delay

Register	Bit	Name	Function	Type	Reset Value
XCR2	1-0	XDATDLY	Transmitter data delay	R/W	00
			XDATDLY = 00		
			0-bit data delay		
			XDATDLY = 01		
			1-bit data delay		
			XDATDLY = 10		
			2-bit data delay		
			XDATDLY = 11		
			Reserved		

20.9.12.1 Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if necessary. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is zero to two bit-clocks (XDATDLY = 00b-10b), as described in [Table 20-59](#) and [Figure 20-56](#). In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-synchronization pulse.

Figure 20-56. Range of Programmable Data Delay


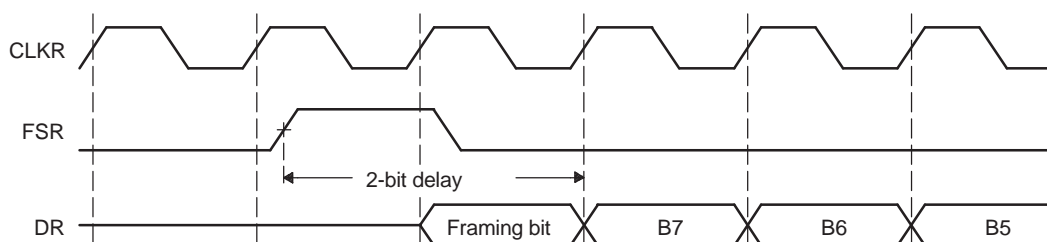
20.9.12.2 0-Bit Data Delay

Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data can be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception this problem is solved because receive data is sampled on the first falling edge of MCLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high and immediately starts driving the first bit to be transmitted on the DX pin.

20.9.12.3 2-Bit Data Delay

A data delay of two bit-periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in the following figure. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 20-57. 2-Bit Data Delay Used to Skip a Framing Bit


20.9.13 Set the Transmit DXENA Mode

Table 20-60. Register Bit Used to Set the Transmit DXENA (DX Delay Enabler) Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR1	7	DXENA	DX delay enabler mode	R/W	0
			DXENA = 0 DX delay enabler is off.		
			DXENA = 1 DX delay enabler is on.		

20.9.13.1 DXENA Mode

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time. This bit does not control the data itself, so only the first bit is delayed.

If you tie together the DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

20.9.14 Set the Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

Table 20-61. Register Bits Used to Set the Transmit Interrupt Mode

Register	Bit	Name	Function	Type	Reset Value
SPCR2	5-4	XINTM	Transmit interrupt mode	R/W	00
			XINTM = 00 XINT generated when XRDY changes from 0 to 1.		
			XINTM = 01 XINT generated by an end-of-block or end-of-frame condition in a transmit multichannel selection mode. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. For details, see Section 20.6.8 . In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.		
			XINTM = 10 XINT generated by a new transmit frame-synchronization pulse. Interrupt on detection of each transmit frame-synchronization pulse. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the CPU clock and sending it to the CPU via XINT.		
			XINTM = 11 XINT generated when XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR can be read to detect this condition. For more information on using XSYNCERR, see Section 20.5.6 .		

20.9.15 Set the Transmit Frame-Synchronization Mode

Table 20-62. Register Bits Used to Set the Transmit Frame-Synchronization Mode

Register	Bit	Name	Function	Type	Reset Value
PCR	11	FSXM	Transmit frame-synchronization mode	R/W	0
			FSXM = 0 Transmit frame synchronization is supplied by an external source via the FSX pin.		
			FSXM = 1 Transmit frame synchronization is supplied by the McBSP, as determined by the FSGM bit of SRGR2.		
SRGR2	12	FSGM	Sample rate generator transmit frame-synchronization mode	R/W	0
			Used when FSXM = 1 in PCR.		
			FSGM = 0 The McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].		
			FSGM = 1 The transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the frame-synchronization period.		

20.9.15.1 Transmit Frame-Synchronization Modes

Table 20-63 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

- External frame-synchronization input
- Sample rate generator frame-synchronization signal (FSG)
- Internal signal that indicates a DXR-to-XSR copy has been made

Table 20-63 also shows the effect of each bit setting on the FSX pin. The polarity of the signal on the FSX pin is determined by the FSXP bit.

Table 20-63. How FSXM and FSGM Select the Source of Transmit Frame-Synchronization Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	FSX Pin Status
0	0 or 1	An external frame-synchronization signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the sample rate generator frame-synchronization signal (FSG).	Output. FSG is inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a transmit frame-synchronization pulse that is 1 cycle wide.	Output. The generated frame-synchronization pulse is inverted as determined by FSXP before being driven out on FSX pin.

20.9.15.2 Other Considerations

If the sample rate generator creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see [Section 20.4.3](#).

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal (SPISTE) on the FSX pin, make sure that FSXM = 1 and FSGM = 0 so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, make sure that FSXM = 0 so that the McBSP can receive the slave-enable signal on the FSX pin.

20.9.16 Set the Transmit Frame-Synchronization Polarity

Table 20-64. Register Bit Used to Set Transmit Frame-Synchronization Polarity

Register	Bit	Name	Function	Type	Reset Value
PCR	3	FSXP	Transmit frame-synchronization polarity	R/W	0
			FSXP = 0 Frame-synchronization pulse FSX is active high.		
			FSXP = 1 Frame-synchronization pulse FSX is active low.		

20.9.16.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be generated internally by the sample rate generator (see [Section 20.4.2](#)) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see [Section 20.9.15](#)). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see [Section 20.9.18](#)).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

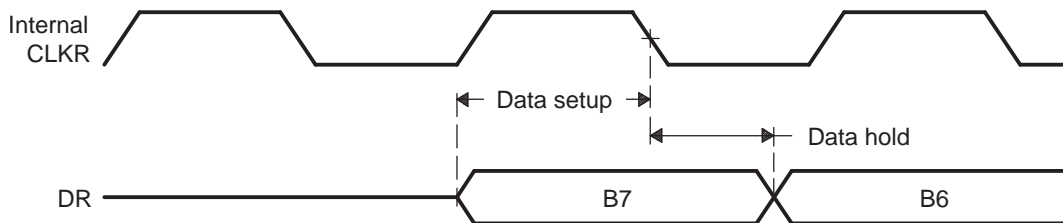
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected and the polarity bit FS(R/X)P = 1, the internal active-high frame-synchronization signals are inverted before being sent to the FS(R/X) pin.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and MCLKR is an input pin), the external rising-edge triggered input clock on MCLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. [Figure 20-58](#) shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 20-58. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



20.9.17 Set the SRG Frame-Synchronization Period and Pulse Width

Table 20-65. Register Bits Used to Set SRG Frame-Synchronization Period and Pulse Width

Register	Bit	Name	Function	Type	Reset Value
SRGR2	11-0	FPER	Sample rate generator frame-synchronization period For the frame-synchronization signal FSG, (FPER + 1) determines the period from the start of a frame-synchronization pulse to the start of the next frame-synchronization pulse. Range for (FPER + 1): 1 to 4096 CLKG cycles.	R/W	0000 0000 0000
SRGR1	15-8	FWID	Sample rate generator frame-synchronization pulse width This field plus 1 determines the width of each frame-synchronization pulse on FSG. Range for (FWID + 1): 1 to 256 CLKG cycles.	R/W	0000 0000

20.9.17.1 Frame-Synchronization Period and Frame-Synchronization Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

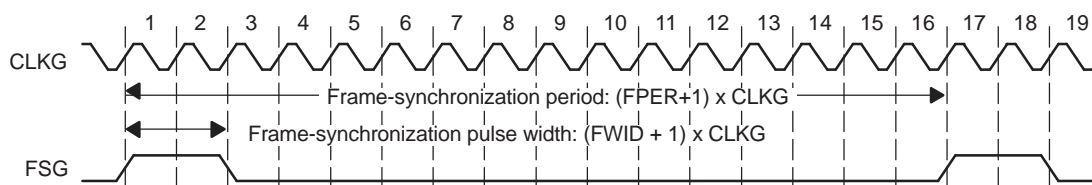
On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 20-59 shows a frame-synchronization period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 20-59. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when GRST = 1 and FSGM = 1, a frame-synchronization pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

20.9.18 Set the Transmit Clock Mode

Table 20-66. Register Bit Used to Set the Transmit Clock Mode

Register	Bit	Name	Function	Type	Reset Value
PCR	9	CLKXM	Transmit clock mode	R/W	0
			CLKXM = 0		The transmitter gets its clock signal from an external source via the MCLKX pin.
			CLKXM = 1		The MCLKX pin is an output pin driven by the sample rate generator of the McBSP.

20.9.18.1 Selecting a Source for the Transmit Clock and a Data Direction for the MCLKX pin

[Table 20-67](#) shows how the CLKXM bit selects the transmit clock and the corresponding status of the MCLKX pin. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit.

Table 20-67. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the MCLKX pin

CLKXM in PCR	Source of Transmit Clock	MCLKX pin Status
0	Internal CLKX is driven by an external clock on the MCLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX.

20.9.18.2 Other Considerations

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the FSR pin. For more details, see [Section 20.4.3](#).

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKXM = 1 so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, make sure that CLKXM = 0 so that CLKX is an input to accept the master clock signal.

20.9.19 Set the Transmit Clock Polarity

Table 20-68. Register Bit Used to Set Transmit Clock Polarity

Register	Bit	Name	Function	Type	Reset Value
PCR	1	CLKXP	Transmit clock polarity	R/W	0
			CLKXP = 0		Transmit data sampled on rising edge of CLKX.
			CLKXP = 1		Transmit data sampled on falling edge of CLKX.

20.9.19.1 Frame Synchronization Pulses, Clock Signals, and Their Polarities

Transmit frame-synchronization pulses can be either generated internally by the sample rate generator (see [Section 20.4.2](#)) or driven by an external source. The source of frame synchronization is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see [Section 20.9.15](#)). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see [Section 20.9.18](#)).

When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP detects them on the internal falling edge of clock, internal MCLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal MCLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, MCLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

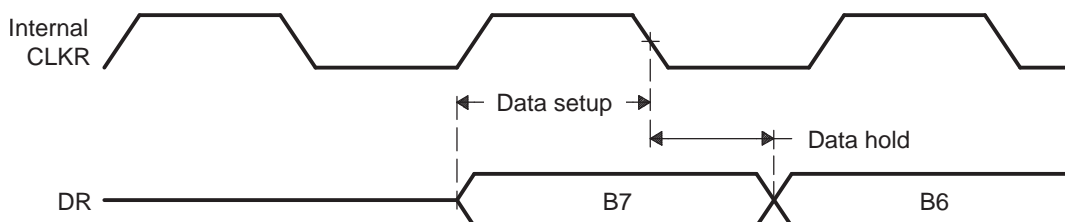
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the MCLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of internal MCLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the MCLKR pin.

CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge (see [Figure 20-58](#)).

[Figure 20-60](#) shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 20-60. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



20.10 Emulation and Reset Considerations

This section covers the following topics:

- How to program McBSP response to a breakpoint in the high-level language debugger (see [Section 20.10.1](#))
- How to reset and initialize the various parts of the McBSP (see [Section 20.10.2](#))

20.10.1 McBSP Emulation Mode

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, the clock continues to run upon a software breakpoint and data is still shifted out. When FREE = 1, the SOFT bit is a *don't care*.

If FREE = 0, the SOFT bit takes effect. If SOFT = 0 when breakpoint occurs, the clock stops immediately, aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer and then the clock halts. These options are listed in [Table 20-69](#).

The McBSP receiver functions in a similar fashion. If a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues running and an overrun error is possible.

Table 20-69. McBSP Emulation Modes Selectable with FREE and SOFT Bits of SPCR2

FREE	SOFT	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition) The transmitter or receiver stops immediately in response to a breakpoint.
0	1	Soft stop mode When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode The transmitter and receiver continue to run when a breakpoint occurs.

20.10.2 Resetting and Initializing McBSPs

20.10.2.1 McBSP Pin States: DSP Reset Versus Receiver/Transmitter Reset

[Table 20-70](#) shows the state of McBSP pins when the serial port is reset due to direct receiver or transmitter reset on the 2833x device.

Table 20-70. Reset State of Each McBSP Pin

Pin	Possible State(s) ⁽¹⁾	State Forced by Device Reset	State Forced by Receiver/Transmitter Reset
Receiver reset (RRST = 0 and GRST = 1)			
MDRx	I	GPIO-input	Input
MCLKRx	I/O/Z	GPIO-input	Known state if input; MCLKR running if output
MFSRx	I/O/Z	GPIO-input	Known state if input; FSRP inactive state if output
Transmitter reset (XRST = 0 and GRST = 1)			
MDXx	O/Z	GPIO Input	High impedance
MCLKXx	I/O/Z	GPIO-input	Known state if input; CLKX running if output
MFSXx	I/O/Z	GPIO-input	Known state if input; FSXP inactive state if output

⁽¹⁾ In Possible State(s) column, I = Input, O = Output, Z = High impedance. In the 28x family, at device reset, all I/Os default to GPIO function and generally as inputs.

20.10.2.2 Device Reset, McBSP Reset, and Sample Rate Generator Reset

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY, XRDY, and XSYNCERR.

- Device reset. When the whole DSP is reset ($\overline{\text{XRS}}$ signal is driven low), all McBSP pins are in GPIO

mode. When the device is pulled out of reset, the clock to the McBSP modules remains disabled.

- **McBSP reset.** When the receiver and transmitter reset bits, RRST and XRST, are loaded with 0s, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. Input-only pins such as MDRx, and all other pins that are configured as inputs are in a known state. The MFSRx and MFSXx pins are driven to their inactive state if they are not outputs. If the MCLKR and MCLKX pins are programmed as outputs, they are driven by CLKG, provided that GRST = 1. Lastly, the MDXx pin is in the high-impedance state when the transmitter and/or the device is reset.

During normal operation, the sample rate generator is reset if the GRST bit is cleared. GRST must be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-synchronization signal (FSG) are driven inactive low.

When the sample rate generator is not in the reset state (GRST = 1), pins MFSRx and MFSXx are in an inactive state when RRST = 0 and XRST = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when GRST = 1 and its frame synchronization is driven by FSG.

- **Sample rate generator reset.** The sample rate generator is reset when GRST is loaded with 0. When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing GRST. In this case, CLKG and FSG are driven inactive low. If you then set GRST, CLKG starts and runs as programmed. Later, if GRST = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.

20.10.2.3 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1. Make XRST = RRST = GRST = 0 in SPCR[1,2]. If coming out of a device reset, this step is not required.
2. While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.
3. Wait for two clock cycles. This ensures proper internal synchronization.
4. Set up data acquisition as required (such as writing to DXR[1,2]).
5. Make XRST = RRST = 1 to enable the serial port. Make sure that as you set these reset bits, you do not modify any of the other bits in SPCR1 and SPCR2. Otherwise, you change the configuration you selected in step 2.
6. Set FRST = 1, if internally generated frame synchronization is required.
7. Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver can be placed in or taken out of reset individually by modifying the desired bit.

The above procedure for reset/initialization can be applied in general when the receiver or transmitter must be reset during its normal operation and when the sample rate generator is not used for either operation.

NOTE:

1. The necessary duration of the active-low period of XRST or RREST is at least two MCLKR/CLKX cycles.
2. The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] must only be modified when the affected portion of the serial port is in its reset state.
3. In most cases, the data transmit registers (DXR[1,2]) must be loaded by the CPU or by the DMA controller only when the transmitter is enabled (XRST = 1). An exception to this rule is when these registers are used for companding internal data (see [Section 20.3.2.2](#)).
4. The bits of the channel control registers—MCR[1,2], RCER[A-H], XCER[A-H]—can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.

20.10.2.4 Resetting the Transmitter While the Receiver is Running

shows values in the control registers that reset and configure the transmitter while the receiver is running.

Example 20-1. Resetting and Configuring McBSP Transmitter While McBSP Receiver Running

```
SPCR1 = 0001h SPCR2 = 0030h
; The receiver is running with the receive interrupt (RINT) triggered by the
; receiver ready bit (RRDY). The transmitter is in its reset state
. The transmit interrupt (XINT) will be triggered by the transmit frame-sync
; error bit (XSYNCERR). PCR = 0900h
; Transmit frame synchronization is generated internally according to the
; FSGM bit of SRGR2.
; The transmit clock is driven by an external source.
; The receive clock continues to be driven by sample rate generator. The input clock
; of the sample rate generator is supplied by the CPU clock SRGR1 = 0001h SRGR2 = 2000h
; The CPU clock is the input clock for the sample rate generator. The sample
; rate generator divides the CPU clock by 2 to generate its output clock (CLKG).
; Transmit frame synchronization is tied to the automatic copying of data from
; the DXR(s) to the XSR(s). XCR1 = 0740h XCR2 = 8321h
; The transmit frame has two phases. Phase 1 has eight 16-bit words. Phase 2
; has four 12-bit words. There is 1-bit data delay between the start of a
; frame-sync pulse and the first data bit
; transmitted. SPCR2 = 0031h
; The transmitter is taken out of reset.
```

20.11 Data Packing Examples

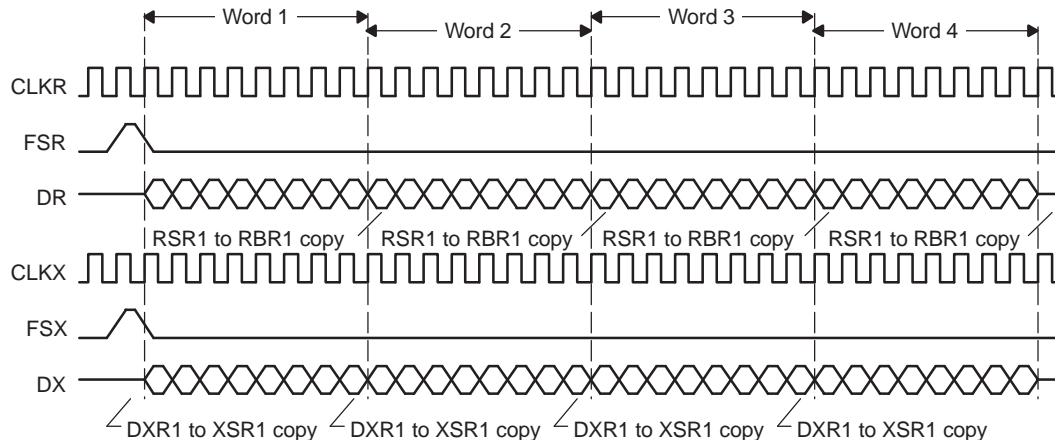
This section shows two ways to implement data packing in the McBSP.

20.11.1 Data Packing Using Frame Length and Word Length

Frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in [Figure 20-61](#). In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the DMA controller. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

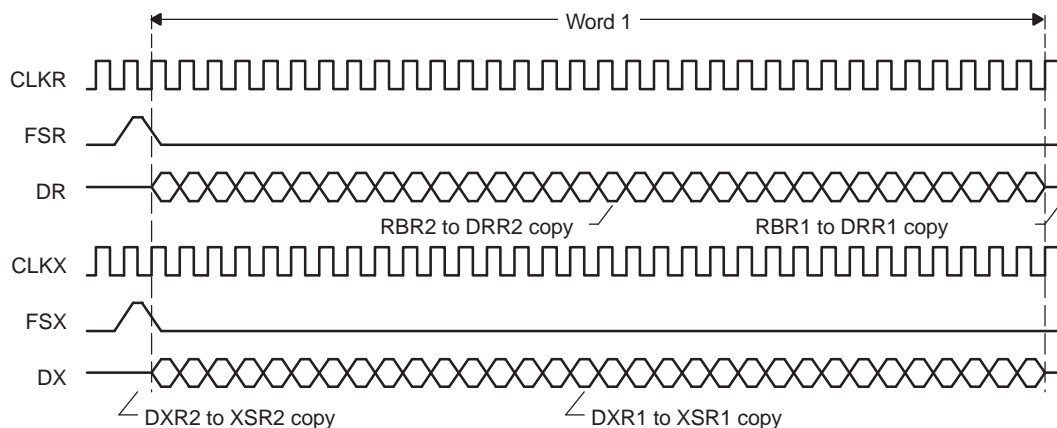
Figure 20-61. Four 8-Bit Data Words Transferred To/From the McBSP


This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 20-62. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or DMA controller. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

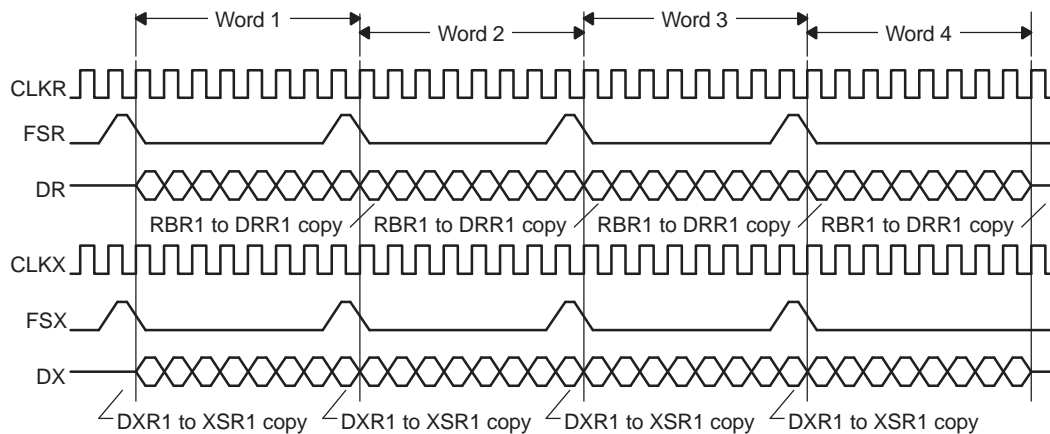
NOTE: When the word length is larger than 16 bits, make sure you access DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

Figure 20-62. One 32-Bit Data Word Transferred To/From the McBSP


20.11.2 Data Packing Using Word Length and the Frame-Synchronization Ignore Function

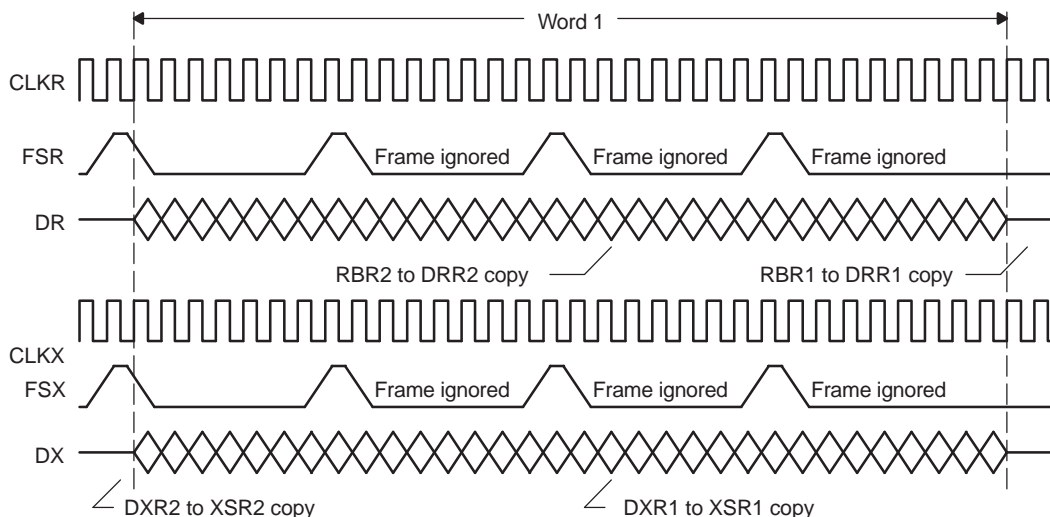
When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-synchronization pulses. First, consider [Figure 20-63](#), which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 8-bit word. Notice the frame-synchronization pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

Figure 20-63. 8-Bit Data Words Transferred at Maximum Packet Frequency



[Figure 20-64](#) shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-synchronization pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 20-64. Configuring the Data Stream of [Figure 20-63](#) as a Continuous 32-Bit Word



20.12 Interrupt Generation

McBSP registers can be programmed to receive and transmit data through DRR2/DRR1 and DXR2/DXR1 registers, respectively. The CPU can directly access these registers to move data from memory to these registers. Interrupt signals will be based on these register pair contents and its related flags. MRINT/MXINT will generate CPU interrupts for receive and transmit conditions.

20.12.1 McBSP Receive Interrupt Generation

In the McBSP module, data receive and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

Figure 20-65. Receive Interrupt Generation

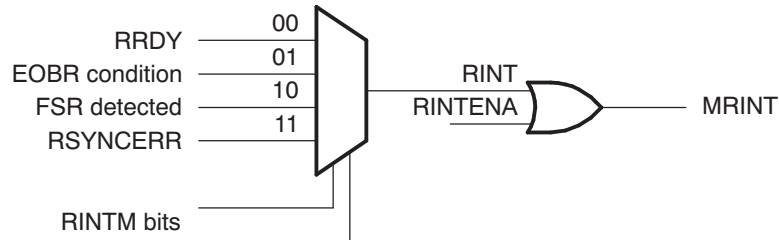


Table 20-71. Receive Interrupt Sources and Signals

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
RINTM Bits					
RINT	RRDY	0	RINTENA	Every word receive	MRINT
	EOBR	1	RINTENA	Every 16 channel block boundary	
	FSR	10	RINTENA	On every FSR	
	RSYNCERR	11	RINTENA	Frame sync error	

NOTE: Since X/RINT, X/REVT, and X/RXFFINT share the same CPU interrupt, it is recommended that all applications use one of the above selections for interrupt generation. If multiple interrupt enables are selected at the same time, there is a likelihood of interrupts being masked or not recognized.

20.12.2 McBSP Transmit Interrupt Generation

McBSP module data transmit and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA.

Figure 20-66. Transmit Interrupt Generation

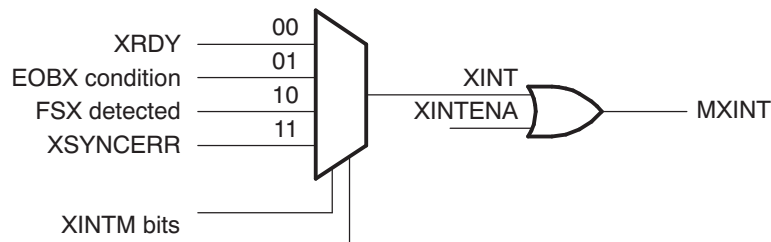


Table 20-72. Transmit Interrupt Sources and Signals

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
XINTM Bits					
XINT	XRDY	0	XINTENA	Every word receive	MXINT
	EOBR	1	XINTENA	Every 16-channel block boundary	

Table 20-72. Transmit Interrupt Sources and Signals (continued)

McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
	FSX	10	XINTENA	On every FSX	
	XSYNCERR	11	XINTENA	Frame sync error	

20.12.3 Error Flags

The McBSP has several error flags both on receive and transmit channel. [Table 20-73](#) explains the error flags and their meaning.

Table 20-73. Error Flags

Error Flags	Function
RFULL	Indicates DRR2/DRR1 are not read and RXR register is overwritten
RSYNCERR	Indicates unexpected frame-sync condition, current data reception will abort and restart. Use RINTM bit 11 for interrupt generation on this condition.
XSYNCERR	Indicates unexpected frame-sync condition, current data transmission will abort and restart. Use XINTM bit 11 for interrupt generation on this condition.

20.12.4 McBSP Interrupt Enable Register

Figure 20-67. McBSP Interrupt Enable Register (MFFINT)

15				8	
Reserved					
R-0					
7	3		2	1	0
Reserved			RINT ENA	Reserved	XINT ENA
R-0			R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-74. McBSP Interrupt Enable Register (MFFINT) Field Descriptions

Bit	Field	Value	Description
15:3	Reserved		Reserved
2	RINT ENA	0 1	Enable for Receive Interrupt Receive interrupt on RRDY is disabled. Receive interrupt on RRDY is enabled.
1	Reserved		
0	XINT ENA	0 1	Enable for transmit Interrupt Transmit interrupt on XRDY is disabled. Transmit interrupt on XRDY is enabled.

20.13 McBSP Modes

McBSP, in its normal mode, communicates with various types of Codecs with variable word size. Apart from this mode, the McBSP uses time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices. The multichannel mode provides flexibility while transmitting/receiving selected channels or all the channels in a TDM stream.

[Table 20-75](#) provides a quick reference to McBSP mode selection.

Table 20-75. McBSP Mode Selection

		Register Bits Used for Mode Selection				
		MCR1 bit 9,0		MCR2 bit 9,1,0		
No.	McBSP Word Size	RMCME	RMCM	XMCME	XMCM	Mode and Function Description
						Normal Mode
1	8/12/16/20/24/32 bit words	0	0	0	0	All types of Codec interface will use this selection
						Multichannel Mode
2	8-bit words					2 Partition or 32-channel Mode
		0	1	0	1	All channels are disabled,unless selected in X/RCERA/B
		0	1	0	10	All channels are enabled,but masked unless selected in X/RCERA/B
		0	1	0	11	Symmetric transmit, receive 8 Partition or 128 Channel Mode Transmit/Receive Channels selected by X/RCERA to X/RCERH bits
						Multichannel Mode is ON
		1	1	1	1	All channels are disabled,unless selected in XCERs
		1	1	1	10	All channels are enabled,but masked unless selected in XCERs
		1	1	1	11	Symmetric transmit, receive
						Continuous Mode - Transmit
		1	0	1	0	Multi-Channel Mode is OFF
						All 128 channels are active and enabled

20.14 McBSP Registers

This section describes the McBSP registers.

20.14.1 McBSP Base Addresses

Table 20-76. McBSP Base Address Table

McbspaRegs	MCBSP_REGS	0x0000_6000	0x0000_603F
McbspbRegs	MCBSP_REGS	0x0000_6040	0x0000_607F

Table 20-77 shows the registers accessible on each McBSP. Section 20.14.2 through Section 20.14.11 describe the register bits.

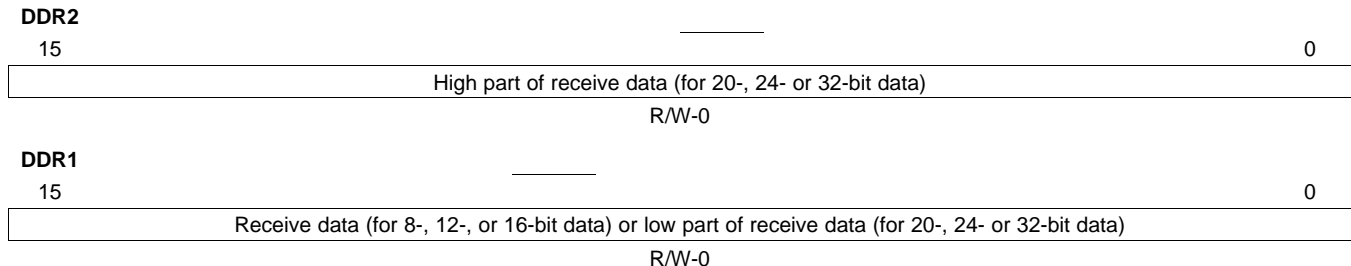
Table 20-77. McBSP Register Summary

Name	McBSP-A Address	McBSP-B Address	Type	Reset Value	Description
Data Registers, Receive, Transmit					
DDR2	0x6000	0x6040	R	0x0000	McBSP Data Receive Register 2
DDR1	0x6001	0x6041	R	0x0000	McBSP Data Receive Register 1
DXR2	0x6002	0x6042	W	0x0000	McBSP Data Transmit Register 2
DXR1	0x6003	0x6043	W	0x0000	McBSP Data Transmit Register 1
McBSP Control Registers					
SPCR2	0x6004	0x6044	R/W	0x0000	McBSP Serial Port Control Register 2
SPCR1	0x6005	0x6045	R/W	0x0000	McBSP Serial Port Control Register 1
RCR2	0x6006	0x6046	R/W	0x0000	McBSP Receive Control Register 2
RCR1	0x6007	0x6047	R/W	0x0000	McBSP Receive Control Register 1
XCR2	0x6008	0x6048	R/W	0x0000	McBSP Transmit Control Register 2
XCR1	0x6009	0x6049	R/W	0x0000	McBSP Transmit Control Register 1
SRGR2	0x600A	0x604A	R/W	0x0000	McBSP Sample Rate Generator Register 2
SRGR1	0x600B	0x604B	R/W	0x0000	McBSP Sample Rate Generator Register 1
Multichannel Control Registers					
MCR2	0x600C	0x604C	R/W	0x0000	McBSP Multichannel Register 2
MCR1	0x600D	0x604D	R/W	0x0000	McBSP Multichannel Register 1
RCERA	0x600E	0x604E	R/W	0x0000	McBSP Receive Channel Enable Register Partition A
RCERB	0x600F	0x604F	R/W	0x0000	McBSP Receive Channel Enable Register Partition B
XCERA	0x6010	0x6050	R/W	0x0000	McBSP Transmit Channel Enable Register Partition A
XCERB	0x6011	0x6051	R/W	0x0000	McBSP Transmit Channel Enable Register Partition B
PCR	0x6012	0x6052	R/W	0x0000	McBSP Pin Control Register
RCERC	0x6013	0x6053	R/W	0x0000	McBSP Receive Channel Enable Register Partition C
RCERD	0x6014	0x6054	R/W	0x0000	McBSP Receive Channel Enable Register Partition D
XCERC	0x6015	0x6055	R/W	0x0000	McBSP Transmit Channel Enable Register Partition C
XCERD	0x6016	0x6056	R/W	0x0000	McBSP Transmit Channel Enable Register Partition D
RCERE	0x6017	0x6057	R/W	0x0000	McBSP Receive Channel Enable Register Partition E
RCERF	0x6018	0x6058	R/W	0x0000	McBSP Receive Channel Enable Register Partition F
XCERE	0x6019	0x6059	R/W	0x0000	McBSP Transmit Channel Enable Register Partition E
XCERF	0x601A	0x605A	R/W	0x0000	McBSP Transmit Channel Enable Register Partition F
RCERG	0x601B	0x605B	R/W	0x0000	McBSP Receive Channel Enable Register Partition G
RCERH	0x601C	0x605C	R/W	0x0000	McBSP Receive Channel Enable Register Partition H
XCERG	0x601D	0x605D	R/W	0x0000	McBSP Transmit Channel Enable Register Partition G
XCERH	0x601E	0x605E	R/W	0x0000	McBSP Transmit Channel Enable Register Partition H
MFFINT	0x6023	0x6063	R/W	0x0000	McBSP Interrupt Enable Register

20.14.2 Data Receive Registers (DDR[1,2])

The CPU or the DMA controller reads received data from one or both of the data receive registers (see [Figure 20-68](#)). If the serial word length is 16 bits or smaller, only DDR1 is used. If the serial length is larger than 16 bits, both DDR1 and DDR2 are used and DDR2 holds the most significant bits. Each frame of receive data in the McBSP can have one phase or two phases, each with its own serial word length.

Figure 20-68. Data Receive Registers (DDR2 and DDR1)



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

20.14.2.1 Data Travel From Data Receive Pins to the Registers

If the serial word length is 16 bits or smaller, receive data on the MDRx pin is shifted into receive shift register 1 (RSR1) and then copied into receive buffer register 1 (RBR1). The content of RBR1 is then copied to DDR1, which can be read by the CPU or by the DMA controller. The RSRs and RBRs are not accessible to the user.

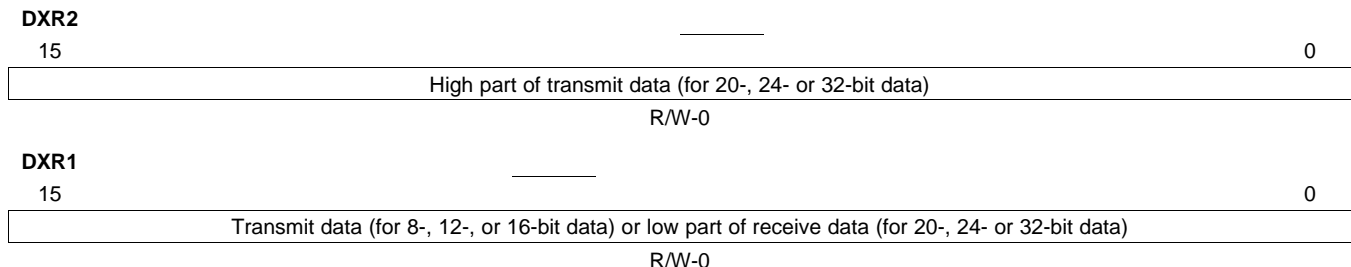
If the serial word length is larger than 16 bits, receive data on the MDRx pin is shifted into both of the receive shift registers (RSR2, RSR1) and then copied into both of the receive buffer registers (RBR2, RBR1). The content of the RBRs is then copied into both of the DRRs, which can be read by the CPU or by the DMA controller.

If companding is used during the copy from RBR1 to DDR1 (RCOMPAND = 10b or 11b), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DDR1. If companding is disabled, the data copied from RBR[1,2] to DDR[1,2] is justified and bit filled according to the RJUST bits.

20.14.3 Data Transmit Registers (DXR[1,2])

For transmission, the CPU or the DMA controller writes data to one or both of the data transmit registers (see [Figure 20-69](#)). If the serial word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, both DXR1 and DXR2 are used and DXR2 holds the most significant bits. Each frame of transmit data in the McBSP can have one phase or two phases, each with its own serial word length.

Figure 20-69. Data Transmit Registers (DXR2 and DXR1)



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

20.14.3.1 Data Travel From Registers to Data Transmit (DX) Pins

If the serial word length is 16 bits or fewer, data written to DXR1 is copied to transmit shift register 1 (XSR1). From XSR1, the data is shifted onto the DX pin one bit at a time. The XSRs are not accessible.

If the serial word length is more than 16 bits, data written to DXR1 and DXR2 is copied to both transmit shift registers (XSR2, XSR1). From the XSRs, the data is shifted onto the DX pin one bit at a time.

If companding is used during the transfer from DXR1 to XSR1 (XCOMPAND = 10b or 11b), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ -law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

20.14.4 Serial Port Control Registers (SPCR[1,2])

Each McBSP has two serial port control registers, SPCR1 (Table 20-78) and SPCR2 (Table 20-79). These registers enable you to:

- Control various McBSP modes: digital loopback mode (DLB), sign-extension and justification mode for reception (RJUST), clock stop mode (CLKSTP), interrupt modes (RINTM and XINTM), emulation mode (FREE and SOFT)
- Turn on and off the DX-pin delay enabler (DXENA)
- Check the status of receive and transmit operations (RSYNCERR, XSYNCERR, RFULL, XEMPTY, RRDY, XRDY)
- Reset portions of the McBSP (RRST, XRST, FRST, GRST)

20.14.4.1 Serial Port Control 1 Register (SPCR1)

The serial port control 1 register (SPCR1) is shown in Figure 20-70 and described in Table 20-78.

Figure 20-70. Serial Port Control 1 Register (SPCR1)

15	14	13	12	11	10	8
DLB	RJUST	CLKSTP	Reserved			
R/W-0	R/W-0	R/W-0	R-0			
7	6	5	4	3	2	1
DXENA	Reserved	RINTM	RSYNCERR	RFULL	RRDY	RRST
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-78. Serial Port Control 1 Register (SPCR1) Field Descriptions

Bit	Field	Value	Description
15	DLB	0	Digital loopback mode bit. DLB disables or enables the digital loopback mode of the McBSP: Disabled
		1	Enabled
			Internal DR is supplied by the MDRx pin. Internal FSR and internal MCLKR can be supplied by their respective pins or by the sample rate generator, depending on the mode bits FSRM and CLKRM.
			Internal DX is supplied by the MDXx pin. Internal FSX and internal CLKX are supplied by their respective pins or are generated internally, depending on the mode bits FSXM and CLKXM.
			Internal receive signals are supplied by internal transmit signals: MDRx connected to MDXx MFSRx connected to MFSXx MCLKR connected to MCLKXx
			This mode allows you to test serial port code with a single DSP. The McBSP transmitter directly supplies data, frame synchronization, and clocking to the McBSP receiver.

Table 20-78. Serial Port Control 1 Register (SPCR1) Field Descriptions (continued)

Bit	Field	Value	Description
14-13	RJUST	0-3h	Receive sign-extension and justification mode bits. During reception, RJUST determines how data is justified and bit filled before being passed to the data receive registers (DRR1, DRR2). RJUST is ignored if you enable a companding mode with the RCOMPAND bits. In a companding mode, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. For more details about the effects of RJUST, see Section 20.8.13 . 0 Right justify the data and zero fill the MSBs. 1h Right justify the data and sign-extend the data into the MSBs. 2h Left justify the data and zero fill the LSBs. 3h Reserved (do not use)
12-11	CLKSTP	0-3h	Clock stop mode bits. CLKSTP allows you to use the clock stop mode to support the SPI master-slave protocol. If you will not be using the SPI protocol, you can clear CLKSTP to disable the clock stop mode. In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). For more details, see Section 20.8.5 . 0-1h Clock stop mode is disabled. 2h Clock stop mode, without clock delay 3h Clock stop mode, with half-cycle clock delay
10-8	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
7	DXENA	0 1	DX delay enabler mode bit. DXENA controls the delay enabler for the DX pin. The enabler creates an extra delay for turn-on time (for the length of the delay, see the device-specific data sheet). For more details about the effects of DXENA, see Section 20.9.13 . 0 DX delay enabler off 1 DX delay enabler on
6	Reserved	0	Reserved
5-4	RINTM	0-3h	Receive interrupt mode bits. RINTM determines which event in the McBSP receiver generates a receive interrupt (RINT) request. If RINT is properly enabled inside the CPU, the CPU services the interrupt request; otherwise, the CPU ignores the request. 0 The McBSP sends a receive interrupt (RINT) request to the CPU when the RRDY bit changes from 0 to 1, indicating that receive data is ready to be read (the content of RBR[1,2] has been copied to DRR[1,2]): Regardless of the value of RINTM, you can check RRDY to determine whether a word transfer is complete. The McBSP sends a RINT request to the CPU when 16 enabled bits have been received on the DR pin. 1h In the multichannel selection mode, the McBSP sends a RINT request to the CPU after every 16-channel block is received in a frame. Outside of the multichannel selection mode, no interrupt request is sent. 2h The McBSP sends a RINT request to the CPU when each receive frame-synchronization pulse is detected. The interrupt request is sent even if the receiver is in its reset state. 3h The McBSP sends a RINT request to the CPU when the RSYNCERR bit is set, indicating a receive frame-synchronization error. Regardless of the value of RINTM, you can check RSYNCERR to determine whether a receive frame-synchronization error occurred.
3	RSYNCERR	0 1	Receive frame-sync error bit. RSYNCERR is set when a receive frame-sync error is detected by the McBSP. If RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU when RSYNCERR is set. The flag remains set until you write a 0 to it or reset the receiver. 0 No error 1 Receive frame-synchronization error. For more details about this error, see Section 20.5.3 .

Table 20-78. Serial Port Control 1 Register (SPCR1) Field Descriptions (continued)

Bit	Field	Value	Description
2	RFULL	0 1	<p>Receiver full bit. RFULL is set when the receiver is full with new data and the previously received data has not been read (receiver-full condition). For more details about this condition, see Section 20.5.2.</p> <p>No receiver-full condition</p> <p>Receiver-full condition: RSR[1,2] and RBR[1,2] are full with new data, but the previous data in DRR[1,2] has not been read.</p>
1	RRDY	0 1	<p>Receiver ready bit. RRDY is set when data is ready to be read from DRR[1,2]. Specifically, RRDY is set in response to a copy from RBR1 to DRR1.</p> <p>If the receive interrupt mode is RINTM = 00b, the McBSP sends a receive interrupt request to the CPU when RRDY changes from 0 to 1.</p> <p>Also, when RRDY changes from 0 to 1, the McBSP sends a receive synchronization event (REVT) signal to the DMA controller.</p> <p>Receiver not ready</p> <p>When the content of DRR1 is read, RRDY is automatically cleared.</p> <p>Receiver ready: New data can be read from DRR[1,2].</p> <p>Important: If both DRRs are required (word length larger than 16 bits), the CPU or the DMA controller must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.</p>
0	RRST	0 1	<p>Receiver reset bit. You can use RRST to take the McBSP receiver into and out of its reset state. This bit has a negative polarity; RRST = 0 indicates the reset state.</p> <p>To read about the effects of a receiver reset, see Section 20.10.2.</p> <p>If you read a 0, the receiver is in its reset state.</p> <p>If you write a 0, you reset the receiver.</p> <p>If you read a 1, the receiver is enabled.</p> <p>If you write a 1, you enable the receiver by taking it out of its reset state.</p>

20.14.4.2 Serial Port Control 2 Register (SPCR2)

The serial port control 2 register (SPCR2) is shown in [Figure 20-71](#) and described in [Table 20-79](#).

Figure 20-71. Serial Port Control 2 Register (SPCR2)

15				10		9	8
Reserved						FREE	SOFT
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-79. Serial Port Control 2 Register (SPCR2) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
9	FREE		Free run bit. When a breakpoint is encountered in the high-level language debugger, FREE determines whether the McBSP transmit and receive clocks continue to run or whether they are affected as determined by the SOFT bit. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.
8	SOFT		Soft stop bit. When FREE = 0, SOFT determines the response of the McBSP transmit and receive clocks when a breakpoint is encountered in the high-level language debugger. When one of the clocks stops, the corresponding data transfer (transmission or reception) stops.
7	FRST	<p>0</p> <p>1</p>	<p>Frame-synchronization logic reset bit. The sample rate generator of the McBSP includes frame-synchronization logic to generate an internal frame-synchronization signal. You can use FRST to take the frame-synchronization logic into and out of its reset state. This bit has a negative polarity; FRST = 0 indicates the reset state.</p> <p>If you read a 0, the frame-synchronization logic is in its reset state.</p> <p>If you write a 0, you reset the frame-synchronization logic.</p> <p>In the reset state, the frame-synchronization logic does not generate a frame-synchronization signal (FSG).</p> <p>If you read a 1, the frame-synchronization logic is enabled.</p> <p>If you write a 1, you enable the frame-synchronization logic by taking it out of its reset state.</p> <p>When the frame-synchronization logic is enabled (FRST = 1) and the sample rate generator as a whole is enabled (GRST = 1), the frame-synchronization logic generates the frame-synchronization signal FSG as programmed.</p>
6	GRST	<p>0</p> <p>1</p>	<p>Sample rate generator reset bit. You can use GRST to take the McBSP sample rate generator into and out of its reset state. This bit has a negative polarity; GRST = 0 indicates the reset state.</p> <p>To read about the effects of a sample rate generator reset, see Section 20.10.2.</p> <p>If you read a 0, the sample rate generator is in its reset state.</p> <p>If you write a 0, you reset the sample rate generator.</p> <p>If GRST = 0 due to a reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven low (inactive). If GRST = 0 due to program code, CLKG and FSG are both driven low (inactive).</p> <p>If you read a 1, the sample rate generator is enabled.</p> <p>If you write a 1, you enable the sample rate generator by taking it out of its reset state.</p> <p>When enabled, the sample rate generator generates the clock signal CLKG as programmed in the sample rate generator registers. If FRST = 1, the generator also generates the frame-synchronization signal FSG as programmed in the sample rate generator registers.</p>

Table 20-79. Serial Port Control 2 Register (SPCR2) Field Descriptions (continued)

Bit	Field	Value	Description
5-4	XINTM	0-3h	Transmit interrupt mode bits. XINTM determines which event in the McBSP transmitter generates a transmit interrupt (XINT) request. If XINT is properly enabled, the CPU services the interrupt request; otherwise, the CPU ignores the request.
		0	The McBSP sends a transmit interrupt (XINT) request to the CPU when the XRDY bit changes from 0 to 1, indicating that transmitter is ready to accept new data (the content of DXR[1,2] has been copied to XSR[1,2]). Regardless of the value of XINTM, you can check XRDY to determine whether a word transfer is complete. The McBSP sends an XINT request to the CPU when 16 enabled bits have been transmitted on the DX pin.
		1h	In the multichannel selection mode, the McBSP sends an XINT request to the CPU after every 16-channel block is transmitted in a frame. Outside of the multichannel selection mode, no interrupt request is sent.
		2h	The McBSP sends an XINT request to the CPU when each transmit frame-synchronization pulse is detected. The interrupt request is sent even if the transmitter is in its reset state.
		3h	The McBSP sends an XINT request to the CPU when the XSYNCERR bit is set, indicating a transmit frame-synchronization error. Regardless of the value of XINTM, you can check XSYNCERR to determine whether a transmit frame-synchronization error occurred.
3	XSYNCERR		Transmit frame-synchronization error bit. XSYNCERR is set when a transmit frame-synchronization error is detected by the McBSP. If XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XSYNCERR is set. The flag remains set until you write a 0 to it or reset the transmitter. If XINTM = 11b, writing a 1 to XSYNCERR triggers a transmit interrupt just as if a transmit frame-synchronization error occurred. For details about this error see Section 20.5.6 .
		0	No error
		1	Transmit frame-synchronization error
2	XEMPTY		Transmitter empty bit. XEMPTY is cleared when the transmitter is ready to send new data but no new data is available (transmitter-empty condition). This bit has a negative polarity; a transmitter-empty condition is indicated by XEMPTY = 0.
		0	Transmitter-empty condition Typically this indicates that all the bits of the current word have been transmitted but there is no new data in DXR1. XEMPTY is also cleared if the transmitter is reset and then restarted. For more details about this error condition, see Section 20.5.5 .
		1	No transmitter-empty condition
1	XRDY		Transmitter ready bit. XRDY is set when the transmitter is ready to accept new data in DXR[1,2]. Specifically, XRDY is set in response to a copy from DXR1 to XSR1. If the transmit interrupt mode is XINTM = 00b, the McBSP sends a transmit interrupt (XINT) request to the CPU when XRDY changes from 0 to 1. Also, when XRDY changes from 0 to 1, the McBSP sends a transmit synchronization event (XEVT) signal to the DMA controller.
		0	Transmitter not ready When DXR1 is loaded, XRDY is automatically cleared.
		1	Transmitter ready: DXR[1,2] is ready to accept new data. If both DXRs are needed (word length larger than 16 bits), the CPU or the DMA controller must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

Table 20-79. Serial Port Control 2 Register (SPCR2) Field Descriptions (continued)

Bit	Field	Value	Description
0	XRST	0	Transmitter reset bit. You can use XRST to take the McBSP transmitter into and out of its reset state. This bit has a negative polarity; XRST = 0 indicates the reset state. To read about the effects of a transmitter reset, see Section 20.10.2 . If you read a 0, the transmitter is in its reset state. If you write a 0, you reset the transmitter.
		1	If you read a 1, the transmitter is enabled. If you write a 1, you enable the transmitter by taking it out of its reset state.

20.14.5 Receive Control Registers (RCR[1, 2])

Each McBSP has two receive control registers, RCR1 ([Table 20-80](#)) and RCR2 ([Table 20-82](#)). These registers enable you to:

- Specify one or two phases for each frame of receive data (RPHASE)
- Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (RWDLEN1, RWDLEN2) and the number of words (RFRLN1, RFRLN2)
- Choose a receive companding mode, if any (RCOMPAND)
- Enable or disable the receive frame-synchronization ignore function (RFIG)
- Choose a receive data delay (RDATDLY)

20.14.5.1 Receive Control Register 1 (RCR1)

The receive control register 1 (RCR1) is shown in [Figure 20-72](#) and described in [Table 20-80](#).

Figure 20-72. Receive Control Register 1 (RCR1)

15	14	8
Reserved	RFRLN1	
R-0	R/W-0	
7	5	4
RWDLEN1	Reserved	0
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-80. Receive Control Register 1 (RCR1) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
14-8	RFRLN1	0-7Fh	Receive frame length 1 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLN1 determines the number of serial words in phase 1 of the frame, and RFRLN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLN fields allow up to 128 words per phase. See Table 20-81 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the RFRLN fields with $[w \text{ minus } 1]$, where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into RFRLN1.

Table 20-80. Receive Control Register 1 (RCR1) Field Descriptions (continued)

Bit	Field	Value	Description
7-5	RWDLEN1	0-7h 0 1h 2h 3h 4h 5h 6h-7h	Receive word length 1. Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame. 8 bits 12 bits 16 bits 20 bits 24 bits 32 bits Reserved (do not use)
4-0	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.

Table 20-81. Frame Length Formula for Receive Control 1 Register (RCR1)

RPHASE	RFRLLEN1	RFRLLEN2	Frame Length
0	$0 \leq \text{RFRLLEN1} \leq 127$	Not used	$(\text{RFRLLEN1} + 1)$ words
1	$0 \leq \text{RFRLLEN1} \leq 127$	$0 \leq \text{RFRLLEN2} \leq 127$	$(\text{RFRLLEN1} + 1) + (\text{RFRLLEN2} + 1)$ words

20.14.5.2 Receive Control Register 2 (RCR2)

The receive control register 2 (RCR2) is shown in [Figure 20-73](#) and described in [Table 20-82](#).

Figure 20-73. Receive Control Register 2 (RCR2)

15	14					8
RPHASE	RFRLLEN2					
R/W-0		R/W-0				
7	5	4	3	2	1	0
RWDLEN2		RCOMPAND		RFIG	RDATDLY	
R/W-0		R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-82. Receive Control Register 2 (RCR2) Field Descriptions

Bit	Field	Value	Description
15	RPHASE	0 1	Receive phase number bit. RPHASE determines whether the receive frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program RWDLEN1 (word length) and RFRLLEN1 (number of words). To set up phase 2 (if there are two phases), program RWDLEN2 and RFRLLEN2. Single-phase frame The receive frame has only one phase, phase 1. Dual-phase frame The receive frame has two phases, phase 1 and phase 2.
14-8		0-7Fh	Receive frame length 2 (1 to 128 words). Each frame of receive data can have one or two phases, depending on value that you load into the RPHASE bit. If a single-phase frame is selected, RFRLLEN1 in RCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, RFRLLEN1 determines the number of serial words in phase 1 of the frame, and RFRLLEN2 in RCR2 determines the number of words in phase 2 of the frame. The 7-bit RFRLLEN fields allow up to 128 words per phase. See Table 20-83 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the RFRLLEN fields with $[w \text{ minus } 1]$, where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 2, load 127 into RFRLLEN2.

Table 20-82. Receive Control Register 2 (RCR2) Field Descriptions (continued)

Bit	Field	Value	Description
7-5	RWDLEN2	0-7h	Receive word length 2. Each frame of receive data can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 in RCR1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 in RCR2 determines the word length in phase 2 of the frame.
		0	8 bits
		1h	12 bits
		2h	16 bits
		3h	20 bits
		4h	24 bits
		5h	32 bits
		6h-7h	Reserved (do not use)
4-3	RCOMPAND	0-3h	Receive companding mode bits. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ -law or A-law format. RCOMPAND allows you to choose one of the following companding modes for the McBSP receiver: For more details about these companding modes, see Section 20.3.2 .
		0	No companding, any size data, MSB received first
		1h	No companding, 8-bit data, LSB received first
		2h	μ -law companding, 8-bit data, MSB received first
		3h	A-law companding, 8-bit data, MSB received first
2	RFIG		Receive frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse. For more details about the frame-synchronization error condition, see Figure 20-30 . Setting RFIG causes the serial port to ignore unexpected frame-synchronization signals during reception. For more details on the effects of RFIG, see Section 20.8.10.1 .
		0	Frame-synchronization detect. An unexpected FSR pulse causes the receiver to discard the contents of RSR[1,2] in favor of the new incoming data. The receiver: 1. Aborts the current data transfer 2. Sets RSYNCERR in SPCR1 3. Begins the transfer of a new data word
		1	Frame-synchronization ignore. An unexpected FSR pulse is ignored. Reception continues uninterrupted.
1-0	RDATDLY	0-3h	Receive data delay bits. RDATDLY specifies a data delay of 0, 1, or 2 receive clock cycles after frame-synchronization and before the reception of the first bit of the frame. For more details, see Section 20.8.12 .
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved (do not use)

Table 20-83. Frame Length Formula for Receive Control 2 Register (RCR2)

RPHASE	RFRLN1	RFRLN2	Frame Length
0	$0 \leq \text{RFRLN1} \leq 127$	Not used	$(\text{RFRLN1} + 1)$ words
1	$0 \leq \text{RFRLN1} \leq 127$	$0 \leq \text{RFRLN2} \leq 127$	$(\text{RFRLN1} + 1) + (\text{RFRLN2} + 1)$ words

20.14.6 Transmit Control Registers (XCR1 and XCR2)

Each McBSP has two transmit control registers, XCR1 ([Table 20-84](#)) and XCR2 ([Table 20-86](#)). These registers enable you to:

- Specify one or two phases for each frame of transmit data (XPHASE)
- Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (XWDLEN1,

XWDLEN2) and the number of words (XFRLEN1, XFRLEN2)

- Choose a transmit companding mode, if any (XCOMPAND)
- Enable or disable the transmit frame-sync ignore function (XFIG)
- Choose a transmit data delay (XDATDLY)

20.14.6.1 Transmit Control 1 Register (XCR1)

The transmit control 1 register (XCR1) is shown in [Figure 20-74](#) and described in [Table 20-84](#).

Figure 20-74. Transmit Control 1 Register (XCR1)

15	14	8
Reserved	XFRLEN1	
R-0	R/W-0	
7	5	4
XWDLEN1	Reserved	
R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-84. Transmit Control 1 Register (XCR1) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reserved bit. Read-only; returns 0 when read.
14-8	XFRLEN1	0-7Fh	Transmit frame length 1 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 20-85 for a summary of how you determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period. Program the XFRLEN fields with [w minus 1], where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.
7-5	XWDLEN1	0-3h	Transmit word length 1. Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame. 0 8 bits 1h 12 bits 2h 16 bits 3h 20 bits 4h 24 bits 5h 32 bits 6h-7h Reserved (do not use)
4-0	Reserved	0	Reserved bits. They are read-only bits and return 0s when read.

Table 20-85. Frame Length Formula for Transmit Control 1 Register (XCR1)

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	$0 \leq \text{XFRLEN1} \leq 127$	Not used	$(\text{XFRLEN1} + 1)$ words
1	$0 \leq \text{XFRLEN1} \leq 127$	$0 \leq \text{XFRLEN2} \leq 127$	$(\text{XFRLEN1} + 1) + (\text{XFRLEN2} + 1)$ words

20.14.6.2 Transmit Control 2 Register (XCR2)

The transmit control 2 register (XCR2) is shown in [Figure 20-75](#) and described in [Table 20-86](#).

Figure 20-75. Transmit Control 2 Register (XCR2)

15	14					8
XPHASE	XFRLEN2					
R/W-0	R/W-0					
7	5	4	3	2	1	0
XWDLEN2		XCOMPAND		XFIG	XDATDLY	
R/W-0		R/W-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-86. Transmit Control 2 Register (XCR2) Field Descriptions

Bit	Field	Value	Description
15	XPHASE	0 1	<p>Transmit phase number bit. XPHASE determines whether the transmit frame has one phase or two phases. For each phase you can define the serial word length and the number of serial words in the phase. To set up phase 1, program XWDLEN1 (word length) and XFRLEN1 (number of words). To set up phase 2 (if there are two phases), program XWDLEN2 and XFRLEN2.</p> <p>0 Single-phase frame The transmit frame has only one phase, phase 1.</p> <p>1 Dual-phase frame The transmit frame has two phases, phase 1 and phase 2.</p>
14-8	XFRLEN2	0-7Fh	<p>Transmit frame length 2 (1 to 128 words). Each frame of transmit data can have one or two phases, depending on value that you load into the XPHASE bit. If a single-phase frame is selected, XFRLEN1 in XCR1 selects the number of serial words (8, 12, 16, 20, 24, or 32 bits per word) in the frame. If a dual-phase frame is selected, XFRLEN1 determines the number of serial words in phase 1 of the frame and XFRLEN2 in XCR2 determines the number of words in phase 2 of the frame. The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 20-87 for a summary of how to determine the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization period.</p> <p>Program the XFRLEN fields with $[w \text{ minus } 1]$, where w represents the number of words per phase. For example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.</p>
7-5	XWDLEN2	0-7h 0 1h 2h 3h 4h 5h 6h-7h	<p>Transmit word length 2. Each frame of transmit data can have one or two phases, depending on the value that you load into the XPHASE bit. If a single-phase frame is selected, XWDLEN1 in XCR1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame and XWDLEN2 in XCR2 determines the word length in phase 2 of the frame.</p> <p>0 8 bits 1h 12 bits 2h 16 bits 3h 20 bits 4h 24 bits 5h 32 bits 6h-7h Reserved (do not use)</p>
4-3	XCOMPAND	0-3h 0 1h 2h 3h	<p>Transmit companding mode bits. Companding (COMPRESS and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. For more details, see Section 20.3.2.</p> <p>XCOMPAND allows you to choose one of the following companding modes for the McBSP transmitter. For more details about these companding modes, see Section 20.3.2.</p> <p>0 No companding, any size data, MSB transmitted first 1h No companding, 8-bit data, LSB transmitted first 2h μ-law companding, 8-bit data, MSB transmitted first 3h A-law companding, 8-bit data, MSB transmitted first</p>

Table 20-86. Transmit Control 2 Register (XCR2) Field Descriptions (continued)

Bit	Field	Value	Description
2	XFIG	0	Transmit frame-synchronization ignore bit. If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse. Setting XFIG causes the serial port to ignore unexpected frame-synchronization pulses during transmission. Frame-synchronization detect. An unexpected FSX pulse causes the transmitter to discard the content of XSR[1,2]. The transmitter: 1. Aborts the present transmission 2. Sets XSYNCERR in SPCR2 3. Begins a new transmission from DXR[1,2]. If new data was written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] is lost. Otherwise, the same data is transmitted.
		1	Frame-synchronization ignore. An unexpected FSX pulse is ignored. Transmission continues uninterrupted.
1-0	XDATDLY	0-3h	Transmit data delay bits. XDADLY specifies a data delay of 0, 1, or 2 transmit clock cycles after frame synchronization and before the transmission of the first bit of the frame. For more details, see Section 20.9.12 .
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved (do not use)

Table 20-87. Frame Length Formula for Transmit Control 2 Register (XCR2)

XPHASE	XFRLN1	XFRLN2	Frame Length
0	$0 \leq \text{XFRLN1} \leq 127$	Not used	$(\text{XFRLN1} + 1)$ words
1	$0 \leq \text{XFRLN1} \leq 127$	$0 \leq \text{XFRLN2} \leq 127$	$(\text{XFRLN1} + 1) + (\text{XFRLN2} + 1)$ words

20.14.7 Sample Rate Generator Registers (SRGR1 and SRGR2)

Each McBSP has two sample rate generator registers, SRGR1 ([Table 20-88](#)) and SRGR2 ([Table 20-89](#)). The sample rate generator can generate a clock signal (CLKG) and a frame-synchronization signal (FSG). The registers SRGR1 and SRGR2 enable you to:

- Select the input clock source for the sample rate generator (CLKSM, in conjunction with the SCLKME bit of PCR)
- Divide down the frequency of CLKG (CLKGDV)
- Select whether internally-generated transmit frame-synchronization pulses are driven by FSG or by activity in the transmitter (FSGM).
- Specify the width of frame-synchronization pulses on FSG (FWID) and specify the period between those pulses (FPER)

When an external source (via the MCLKR or MCLKX pin) provides the input clock source for the sample rate generator:

- If the CLKX/MCLKR pin is used, the polarity of the input clock is selected with CLKXP/CLKRP of PCR.
- The GSYNC bit of SRGR2 allows you to make CLKG synchronized to an external frame-synchronization signal on the FSR pin, so that CLKG is kept in phase with the input clock.

20.14.7.1 Sample Rate Generator 1 Register (SRGR1)

The sample rate generator 1 register is shown in [Figure 20-76](#) and described in [Table 20-88](#).

Figure 20-76. Sample Rate Generator 1 Register (SRGR1)

15	8
FWID	
R/W-0	
7	0
CLKGDV	
R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-88. Sample Rate Generator 1 Register (SRGR1) Field Descriptions

Bit	Field	Value	Description															
15-8	FWID	0-FFh	<p>Frame-synchronization pulse width bits for FSG</p> <p>The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. For frame-synchronization pulses on FSG, (FWID + 1) is the pulse width in CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles:</p> $0 \leq \text{FWID} \leq 255$ $1 \leq (\text{FWID} + 1) \leq 256 \text{ CLKG cycles}$ <p>The period between the frame-synchronization pulses on FSG is defined by the FPER bits.</p>															
7-0	CLKGDV	0-FFh	<p>Divide-down value for CLKG. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:</p> $\text{CLKG frequency} = (\text{Input clock frequency})/ (\text{CLKGDV} + 1)$ <p>The input clock is selected by the SCLKME and CLKSM bits:</p> <table><thead><tr><th>SCLKME</th><th>CLKSM</th><th>Input Clock For Sample Rate Generator</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>LSPCLK</td></tr><tr><td>1</td><td>0</td><td>Signal on MCLKR pin</td></tr><tr><td>1</td><td>1</td><td>Signal on MCLKX pin</td></tr></tbody></table>	SCLKME	CLKSM	Input Clock For Sample Rate Generator	0	0	Reserved	0	1	LSPCLK	1	0	Signal on MCLKR pin	1	1	Signal on MCLKX pin
SCLKME	CLKSM	Input Clock For Sample Rate Generator																
0	0	Reserved																
0	1	LSPCLK																
1	0	Signal on MCLKR pin																
1	1	Signal on MCLKX pin																

20.14.7.2 Sample Rate Generator 2 Register (SRGR2)

The sample rate generator 2 register (SRGR2) is shown in [Figure 20-77](#) and described in [Table 20-89](#).

Figure 20-77. Sample Rate Generator 2 Register (SRGR2)

15	14	13	12	11	8
GSYNC	Reserved	CLKSM	FSGM	FPER	
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	
7					0
FPER					
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-89. Sample Rate Generator 2 Register (SRGR2) Field Descriptions

Bit	Field	Value	Description																		
15	GSYNC	<div>0</div> <div>1</div>	<p>Clock synchronization mode bit for CLKG. GSYNC is used only when the input clock source for the sample rate generator is external—on the MCLKR pin.</p> <p>When GSYNC = 1, the clock signal (CLKG) and the frame-synchronization signal (FSG) generated by the sample rate generator are made dependent on pulses on the FSR pin.</p> <p>No clock synchronization</p> <p>CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.</p> <p>Clock synchronization</p> <ul style="list-style-type: none">• CLKG is adjusted as necessary so that it is synchronized with the input clock on the MCLKR pin.• FSG pulses. FSG only pulses in response to a pulse on the FSR pin. <p>The frame-synchronization period defined in FPER is ignored.</p> <p>For more details, see Section 20.4.3.</p>																		
14	Reserved		Reserved																		
13	CLKSM	<div>0</div> <div>1</div>	<p>Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:)</p> <p>CLKG frequency = (input clock frequency)/ (CLKGDV + 1</p> <p>CLKSM is used in conjunction with the SCLKME bit to determine the source for the input clock.</p> <p>A reset selects the CPU clock as the input clock and forces the CLKG frequency to ½ the LSPCLK frequency.</p> <p>The input clock for the sample rate generator is taken from the MCLKR pin, depending on the value of the SCLKME bit of PCR:</p> <table><thead><tr><th>SCLKME</th><th>CLKSM</th><th>Input Clock For Sample Rate Generator</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>0</td><td>Signal on MCLKR pin</td></tr></tbody></table> <p>The input clock for the sample rate generator is taken from the LSPCLK or from the MCLKX pin, depending on the value of the SCLKME bit of PCR:</p> <table><thead><tr><th>SCLKME</th><th>CLKSM</th><th>Input Clock For Sample Rate Generator</th></tr></thead><tbody><tr><td>0</td><td>1</td><td>LSPCLK</td></tr><tr><td>1</td><td>1</td><td>Signal on MCLKX pin</td></tr></tbody></table>	SCLKME	CLKSM	Input Clock For Sample Rate Generator	0	0	Reserved	1	0	Signal on MCLKR pin	SCLKME	CLKSM	Input Clock For Sample Rate Generator	0	1	LSPCLK	1	1	Signal on MCLKX pin
SCLKME	CLKSM	Input Clock For Sample Rate Generator																			
0	0	Reserved																			
1	0	Signal on MCLKR pin																			
SCLKME	CLKSM	Input Clock For Sample Rate Generator																			
0	1	LSPCLK																			
1	1	Signal on MCLKX pin																			
12	FSGM	<div>0</div> <div>1</div>	<p>Sample rate generator transmit frame-synchronization mode bit. The transmitter can get frame synchronization from the FSX pin (FSXM = 0) or from inside the McBSP (FSXM = 1). When FSXM = 1, the FSGM bit determines how the McBSP supplies frame-synchronization pulses.</p> <p>If FSXM = 1, the McBSP generates a transmit frame-synchronization pulse when the content of DXR[1,2] is copied to XSR[1,2].</p> <p>If FSXM = 1, the transmitter uses frame-synchronization pulses generated by the sample rate generator. Program the FWID bits to set the width of each pulse. Program the FPER bits to set the period between pulses.</p>																		
11-0	FPER	0-FFFh	<p>Frame-synchronization period bits for FSG. The sample rate generator can produce a clock signal, CLKG, and a frame-synchronization signal, FSG. The period between frame-synchronization pulses on FSG is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles:</p> <p>0 ≤ FPER ≤ 4095</p> <p>1 ≤ (FPER + 1) ≤ 4096 CLKG cycles</p> <p>The width of each frame-synchronization pulse on FSG is defined by the FWID bits.</p>																		

20.14.8 Multichannel Control Registers (MCR[1,2])

Each McBSP has two multichannel control registers. MCR1 ([Table 20-90](#)) has control and status bits (with an R prefix) for multichannel selection operation in the receiver. MCR2 ([Table 20-91](#)) contains the same type of bits (bit with an X prefix) for the transmitter. These registers enable you to:

Table 20-90. Multichannel Control 1 Register (MCR1) Field Descriptions (continued)

Bit	Field	Value	Description
8-7	RPBBLK	0-3h	<p>Receive partition B block bits</p> <p>RPBBLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver.</p> <p>The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use RPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B. Use the RPABLK bits to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.</p> <p>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLK bits are regularly updated to indicate which block is active.</p> <p>When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).</p> <p>0 Block 1: channels 16 through 31 1h Block 3: channels 48 through 63 2h Block 5: channels 80 through 95 3h Block 7: channels 112 through 127</p>
6-5	RPABLK	0-3h	<p>Receive partition A block bits</p> <p>RPABLK is only applicable if channels can be individually enabled or disabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the description for RPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.</p> <p>0 Block 0: channels 0 through 15 1h Block 2: channels 32 through 47 2h Block 5: channels 64 through 79 3h Block 7: channels 96 through 111</p>
4-2	RCBLK	0-7h	<p>Receive current block indicator. RCBLK indicates which block of 16 channels is involved in the current McBSP reception:</p> <p>0 Block 0: channels 0 through 15 1h Block 1: channels 16 through 31 2h Block 2: channels 32 through 47 3h Block 3: channels 48 through 63 4h Block 4: channels 64 through 79 5h Block 5: channels 80 through 95 6h Block 6: channels 96 through 111 7h Block 7: channels 112 through 127</p>
1	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
0	RMCM	0 1	<p>Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception:</p> <p>0 All 128 channels are enabled. 1 Multichanneled selection mode. Channels can be individually enabled or disabled.</p> <p>The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.</p>

20.14.8.2 Multichannel Control 2 Register (MCR2)

The multichannel control 2 register (MCR2) is shown in [Figure 20-79](#) and described in [Table 20-91](#).

Figure 20-79. Multichannel Control 2 Register (MCR2)

15								10		9		8
Reserved										XMCME		XPBBLK
R-0										R/W-0		R/W-0
7	6	5	4					2		1		0
XPBBLK	XPABLK			XCBLK				XMCM				
R/W-0	R/W-0			R-0				R/W-0				

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-91. Multichannel Control 2 Register (MCR2) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved bits (not available for your use). They are read-only bits and return 0s when read.
9	XMCME	0	Transmit multichannel partition mode bit. XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). 2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b(for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A XCERB: Channels in partition B
		1	8-partition mode. All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels 0 through 15 XCERB: Channels 16 through 31 XCERC: Channels 32 through 47 XCERD: Channels 48 through 63 XCERE: Channels 64 through 79 XCERF: Channels 80 through 95 XCERG: Channels 96 through 111 XCERH: Channels 112 through 127

Table 20-91. Multichannel Control 2 Register (MCR2) Field Descriptions (continued)

Bit	Field	Value	Description																
8-7	XPBBLK	0-3h	<p>Transmit partition B block bits</p> <p>XPBBLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter.</p> <p>The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use XPBBLK to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B, as shown in the following table. Use the XPABLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.</p> <p>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bits are regularly updated to indicate which block is active.</p> <p>When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).</p> <table><tr><td>0</td><td>Block 1: channels 16 through 31</td></tr><tr><td>1h</td><td>Block 3: channels 48 through 63</td></tr><tr><td>2h</td><td>Block 5: channels 80 through 95</td></tr><tr><td>3h</td><td>Block 7: channels 112 through 127</td></tr></table>	0	Block 1: channels 16 through 31	1h	Block 3: channels 48 through 63	2h	Block 5: channels 80 through 95	3h	Block 7: channels 112 through 127								
0	Block 1: channels 16 through 31																		
1h	Block 3: channels 48 through 63																		
2h	Block 5: channels 80 through 95																		
3h	Block 7: channels 112 through 127																		
6-5	XPABLK	<table><tr><td>0</td><td>Block 0: channels 0 through 15</td></tr><tr><td>1h</td><td>Block 2: channels 32 through 47</td></tr><tr><td>2h</td><td>Block 4: channels 64 through 79</td></tr><tr><td>3h</td><td>Block 6: channels 96 through 111</td></tr></table>	0	Block 0: channels 0 through 15	1h	Block 2: channels 32 through 47	2h	Block 4: channels 64 through 79	3h	Block 6: channels 96 through 111	<p>Transmit partition A block bits. XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. See the description for XPBBLK (bits 8-7) for more information about assigning blocks to partitions A and B.</p>								
0	Block 0: channels 0 through 15																		
1h	Block 2: channels 32 through 47																		
2h	Block 4: channels 64 through 79																		
3h	Block 6: channels 96 through 111																		
4-2	XCBLK	<table><tr><td>0</td><td>Block 0: channels 0 through 15</td></tr><tr><td>1h</td><td>Block 1: channels 16 through 31</td></tr><tr><td>2h</td><td>Block 2: channels 32 through 47</td></tr><tr><td>3h</td><td>Block 3: channels 48 through 63</td></tr><tr><td>4h</td><td>Block 4: channels 64 through 79</td></tr><tr><td>5h</td><td>Block 5: channels 80 through 95</td></tr><tr><td>6h</td><td>Block 6: channels 96 through 111</td></tr><tr><td>7h</td><td>Block 7: channels 112 through 127</td></tr></table>	0	Block 0: channels 0 through 15	1h	Block 1: channels 16 through 31	2h	Block 2: channels 32 through 47	3h	Block 3: channels 48 through 63	4h	Block 4: channels 64 through 79	5h	Block 5: channels 80 through 95	6h	Block 6: channels 96 through 111	7h	Block 7: channels 112 through 127	<p>Transmit current block indicator. XCBLK indicates which block of 16 channels is involved in the current McBSP transmission:</p>
0	Block 0: channels 0 through 15																		
1h	Block 1: channels 16 through 31																		
2h	Block 2: channels 32 through 47																		
3h	Block 3: channels 48 through 63																		
4h	Block 4: channels 64 through 79																		
5h	Block 5: channels 80 through 95																		
6h	Block 6: channels 96 through 111																		
7h	Block 7: channels 112 through 127																		
1-0	XMCM	0-3h	<p>Transmit multichannel selection mode bits. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission. For more details on how the channels are affected, see Section 20.6.7.</p> <table><tr><td>0</td><td>No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.</td></tr><tr><td>1h</td><td>All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.</td></tr><tr><td>2h</td><td>All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).</td></tr><tr><td>3h</td><td>This mode is used for symmetric transmission and reception.</td></tr></table> <p>All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs).</p> <p>The XMCM bit determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.</p>	0	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.	1h	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.	2h	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).	3h	This mode is used for symmetric transmission and reception.								
0	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.																		
1h	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs). If enabled, a channel in this mode is also unmasked.																		
2h	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs).																		
3h	This mode is used for symmetric transmission and reception.																		

20.14.9 Pin Control Register (PCR)

Each McBSP has one pin control register (PCR). [Table 20-92](#) describes the bits of PCR. This register enables you to:

- Choose a frame-synchronization mode for the transmitter (FSXM) and for the receiver (FSRM)
- Choose a clock mode for transmitter (CLKXM) and for the receiver (CLKRM)
- Select the input clock source for the sample rate generator (SCLKME, in conjunction with the CLKSM bit of SRGR2)
- Choose whether frame-synchronization signals are active low or active high (FSXP for transmission, FSRP for reception)
- Specify whether data is sampled on the falling edge or the rising edge of the clock signals (CLKXP for transmission, CLKRP for reception)

The pin control register (PCR) is shown in [Figure 20-80](#) and described in [Table 20-92](#).

Figure 20-80. Pin Control Register (PCR)

15	12	11	10	9	8
Reserved		FSXM	FSRM	CLKXM	CLKRM
R-0		R/W-0	R/W-0	R/W-0	R/W-0
7	6	4	3	2	1
SCLKME	Reserved	FSXP	FSRP	CLKXP	CLKRP
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-92. Pin Control Register (PCR) Field Descriptions

Bit	Field	Value	Description
15:12	Reserved	0	Reserved bit (not available for your use). It is a read-only bit and returns a 0 when read.
11	FSXM	0	Transmit frame-synchronization mode bit. FSXM determines whether transmit frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSX pin is determined by the FSXP bit.
		1	Transmit frame synchronization is generated internally by the Sample Rate generator, as determined by the FSGM bit of SRGR2.
10	FSRM	0	Receive frame-synchronization mode bit. FSRM determines whether receive frame-synchronization pulses are supplied externally or internally. The polarity of the signal on the FSR pin is determined by the FSRP bit.
		1	Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.

Table 20-92. Pin Control Register (PCR) Field Descriptions (continued)

Bit	Field	Value	Description																		
9	CLKXM	<div>0</div> <div>1</div>	<p>Transmit clock mode bit. CLKXM determines whether the source for the transmit clock is external or internal, and whether the MCLKX pin is an input or an output. The polarity of the signal on the MCLKX pin is determined by the CLKXP bit.</p> <p>In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKX is an output. If the McBSP is a slave, make sure that CLKX is an input.</p> <p>Not in clock stop mode (CLKSTP = 00b or 01b):</p> <p>The transmitter gets its clock signal from an external source via the MCLKX pin.</p> <p>Internal CLKX is driven by the sample rate generator of the McBSP. The MCLKX pin is an output pin that reflects internal CLKX.</p> <p>In clock stop mode (CLKSTP = 10b or 11b):</p> <div><div>0</div><div>1</div></div> <p>The McBSP is a slave in the SPI protocol. The internal transmit clock (CLKX) is driven by the SPI master via the MCLKX pin. The internal receive clock (MCLKR) is driven internally by CLKX, so that both the transmitter and the receiver are controlled by the external master clock.</p> <p>The McBSP is a master in the SPI protocol. The sample rate generator drives the internal transmit clock (CLKX). Internal CLKX is reflected on the MCLKX pin to drive the shift clock of the SPI-compliant slaves in the system. Internal CLKX also drives the internal receive clock (MCLKR), so that both the transmitter and the receiver are controlled by the internal master clock.</p>																		
8	CLKRM	<div>0</div> <div>1</div> <div>0</div> <div>1</div>	<p>Receive clock mode bit. The role of CLKRM and the resulting effect on the MCLKR pin depend on whether the McBSP is in the digital loopback mode (DLB = 1).</p> <p>The polarity of the signal on the MCLKR pin is determined by the CLKRP bit.</p> <p>Not in digital loopback mode (DLB = 0):</p> <p>The MCLKR pin is an input pin that supplies the internal receive clock (MCLKR).</p> <p>Internal MCLKR is driven by the sample rate generator of the McBSP. The MCLKR pin is an output pin that reflects internal MCLKR.</p> <p>In digital loopback mode (DLB = 1):</p> <div><div>0</div><div>1</div></div> <p>The MCLKR pin is in the high impedance state. The internal receive clock (MCLKR) is driven by the internal transmit clock (CLKX). CLKX is derived according to the CLKXM bit.</p> <p>Internal MCLKR is driven by internal CLKX. The MCLKR pin is an output pin that reflects internal MCLKR. CLKX is derived according to the CLKXM bit.</p>																		
7	SCLKME		<p>Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is:</p> $\text{CLKG freq.} = (\text{Input clock frequency}) / (\text{CLKGDV} + 1)$ <p>SCLKME is used in conjunction with the CLKSM bit to select the input clock.</p> <table><thead><tr><th>SCLKME</th><th>CLKSM</th><th>Input Clock For Sample Rate Generator</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>LSPCLK</td></tr></tbody></table> <p>The input clock for the sample rate generator is taken from the MCLKR pin or from the MCLKX pin, depending on the value of the CLKSM bit of SRGR2:</p> <table><thead><tr><th>SCLKME</th><th>CLKSM</th><th>Input Clock For Sample Rate Generator</th></tr></thead><tbody><tr><td>1</td><td>0</td><td>Signal on MCLKR pin</td></tr><tr><td>1</td><td>1</td><td>Signal on MCLKX pin</td></tr></tbody></table>	SCLKME	CLKSM	Input Clock For Sample Rate Generator	0	0	Reserved	0	1	LSPCLK	SCLKME	CLKSM	Input Clock For Sample Rate Generator	1	0	Signal on MCLKR pin	1	1	Signal on MCLKX pin
SCLKME	CLKSM	Input Clock For Sample Rate Generator																			
0	0	Reserved																			
0	1	LSPCLK																			
SCLKME	CLKSM	Input Clock For Sample Rate Generator																			
1	0	Signal on MCLKR pin																			
1	1	Signal on MCLKX pin																			
6-4	Reserved		Reserved																		
3	FSXP	<div>0</div> <div>1</div>	<p>Transmit frame-synchronization polarity bit. FSXP determines the polarity of FSX as seen on the FSX pin.</p> <p>Transmit frame-synchronization pulses are active high.</p> <p>Transmit frame-synchronization pulses are active low.</p>																		
2	FSRP	<div>0</div> <div>1</div>	<p>Receive frame-synchronization polarity bit. FSRP determines the polarity of FSR as seen on the FSR pin.</p> <p>Receive frame-synchronization pulses are active high.</p> <p>Receive frame-synchronization pulses are active low.</p>																		

Table 20-92. Pin Control Register (PCR) Field Descriptions (continued)

Bit	Field	Value	Description
1	CLKXP	0	Transmit clock polarity bit. CLKXP determines the polarity of CLKX as seen on the MCLKX pin. Transmit data is sampled on the rising edge of CLKX.
		1	Transmit data is sampled on the falling edge of CLKX.
0	CLKRP	0	Receive clock polarity bit. CLKRP determines the polarity of CLKR as seen on the MCLKR pin. Receive data is sampled on the falling edge of MCLKR.
		1	Receive data is sampled on the rising edge of MCLKR.

Table 20-93. Pin Configuration

Pin	Selected as Output When ...	Selected as Input When ...
CLKX	CLKXM = 1	CLKXM = 0
FSX	FSXM = 1	FSXM = 0
CLKR	CLKRM = 1	CLKRM = 0
FSR	FSRM = 1	FSRM = 0

20.14.10 Receive Channel Enable Registers (RCERA, RCERB, RCERC, RCERD, RCERE, RCERF, RCERG, RCERH)

Each McBSP has eight receive channel enable registers of the format shown in [Figure 20-81](#). There is one enable register for each of the receive partitions: A, B, C, D, E, F, G, and H. [Table 20-94](#) provides a summary description that applies to any bit x of a receive channel enable register.

These memory-mapped registers are only used when the receiver is configured to allow individual enabling and disabling of the channels (RMCM = 1). For more details about the way these registers are used, see [Section 20.14.10.1](#).

The receive channel enable registers (RCERA...RCERH) are shown in [Figure 20-81](#) and described in [Table 20-94](#).

Figure 20-81. Receive Channel Enable Registers (RCERA...RCERH)

15	14	13	12	11	10	9	8
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RCE7	RCE6	RCE5	RCE4	RCE3	RCE2	RCE1	RCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-94. Receive Channel Enable Registers (RCERA...RCERH) Field Descriptions

Bit	Field	Value	Description
15-0	RCEx		Receive channel enable bit.
			For receive multichannel selection mode (RMCM = 1):
		0	Disable the channel that is mapped to RCEx.
		1	Enable the channel that is mapped to RCEx.

20.14.10.1 RCERs Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the RCERs depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit. For each of these two cases, [Table 20-95](#) shows which block of channels is assigned to each of the RCERs used. For each RCER, the table shows which channel is assigned to each of the bits.

Table 20-95. Use of the Receive Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
32 (RMCME = 0)	RCERA	Channels n to (n + 15) The block of channels is chosen with the RPABLK bits.	RCE0	Channel n
			RCE1	Channel (n + 1)
			RCE2	Channel (n + 2)
			:	:
			RCE15	Channel (n + 15)
	RCERB	Channels m to (m + 15) The block of channels is chosen with the RPBBLK bits.	RCE0	Channel m
			RCE1	Channel (m + 1)
			RCE2	Channel (m + 2)
			:	:
			RCE15	Channel (m + 15)
128 (RMCME = 1)	RCERA	Block 0	RCE0	Channel 0
			RCE1	Channel 1
			RCE2	Channel 2
			:	:
			RCE15	Channel 15
	RCERB	Block 1	RCE0	Channel 16
			RCE1	Channel 17
			RCE2	Channel 18
			:	:
			RCE15	Channel 31
	RCERC	Block 2	RCE0	Channel 32
			RCE1	Channel 33
			RCE2	Channel 34
			:	:
			RCE15	Channel 47
	RCERD	Block 3	RCE0	Channel 48
			RCE1	Channel 49
			RCE2	Channel 50
			:	:
			RCE15	Channel 63
	RCERE	Block 4	RCE0	Channel 64
			RCE1	Channel 65
			RCE2	Channel 66
			:	:
			RCE15	Channel 79
	RCERF	Block 5	RCE0	Channel 80
			RCE1	Channel 81
			RCE2	Channel 82
			:	:
			RCE15	Channel 95
	RCERG	Block 6	RCE0	Channel 96
			RCE1	Channel 97
			RCE2	Channel 98
			:	:
			RCE15	Channel 111

Table 20-95. Use of the Receive Channel Enable Registers (continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	RCERx	Block Assigned	Bit in RCERx	Channel Assigned
	RCERH	Block 7	RCE0	Channel 112
			RCE1	Channel 113
			RCE2	Channel 114
			:	:
			RCE15	Channel 127

20.14.11 Transmit Channel Enable Registers (XCERA, XCERB, XCERC, XCERD, XCERE, XCERF, XCERG, X CERH)

Each McBSP has eight transmit channel enable registers of the form shown in Figure 20-82. There is one for each of the transmit partitions: A, B, C, D, E, F, G, and H. Table 20-96 provides a summary description that applies to each bit XCE_x of a transmit channel enable register.

The XCERs are only used when the transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (XMCM is nonzero).

The transmit channel enable registers (XCERA...XCERH) are shown in Figure 20-82 and described in Table 20-96.

Figure 20-82. Transmit Channel Enable Registers (XCERA...XCERH)

15	14	13	12	11	10	9	8
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XCE7	XCE6	XCE5	XCE4	XCE3	XCE2	XCE1	XCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 20-96. Transmit Channel Enable Registers (XCERA...XCERH) Field Descriptions

Bit	Field	Value	Description
15-0	XCE _x		Transmit channel enable bit. The role of this bit depends on which transmit multichannel selection mode is selected with the XMCM bits.
			For multichannel selection when XMCM = 01b (all channels disabled unless selected):
		0	Disable and mask the channel that is mapped to XCE _x .
		1	Enable and unmask the channel that is mapped to XCE _x .
			For multichannel selection when XMCM = 10b (all channels enabled but masked unless selected):
		0	Mask the channel that is mapped to XCE _x .
		1	Unmask the channel that is mapped to XCE _x .
			For multichannel selection when XMCM = 11b (all channels masked unless selected):
		0	Mask the channel that is mapped to XCE _x . Even if the channel is enabled by the corresponding receive channel enable bit, this channel's data cannot appear on the DX pin.
		1	Unmask the channel that is mapped to XCE _x . If the channel is also enabled by the corresponding receive channel enable bit, full transmission can occur.

20.14.11.1 XCERs Used in a Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit. These two cases are shown in [Table 20-97](#). The table shows which block of channels is assigned to each XCER that is used. For each XCER, the table shows which channel is assigned to each of the bits.

NOTE: When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

Table 20-97. Use of the Transmit Channel Enable Registers

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	Block Assigned	Bit in XCERx	Channel Assigned
32 (XMCME = 0)	XCERA	Channels n to (n + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPABLK bits. When XMCM = 11b, the block is chosen with the RPABLK bits.	XCE0	Channel n
			XCE1	Channel (n + 1)
			XCE2	Channel (n + 2)
			:	:
			XCE15	Channel (n + 15)
	XCERB	Channels m to (m + 15) When XMCM = 01b or 10b, the block of channels is chosen with the XPBBLK bits. When XMCM = 11b, the block is chosen with the RPBBLK bits.	XCE0	Channel m
			XCE1	Channel (m + 1)
			XCE2	Channel (m + 2)
			:	:
			XCE15	Channel (m + 15)
128 (XMCME = 1)	XCERA	Block 0	XCE0	Channel 0
			XCE1	Channel 1
			XCE2	Channel 2
			:	:
			XCE15	Channel 15
	XCERB	Block 1	XCE0	Channel 16
			XCE1	Channel 17
			XCE2	Channel 18
			:	:
			XCE15	Channel 31
	XCERC	Block 2	XCE0	Channel 32
			XCE1	Channel 33
			XCE2	Channel 34
			:	:
			XCE15	Channel 47
	XCERD	Block 3	XCE0	Channel 48
			XCE1	Channel 49
			XCE2	Channel 50
			:	:
			XCE15	Channel 63

Table 20-97. Use of the Transmit Channel Enable Registers (continued)

Number of Selectable Channels	Block Assignments		Channel Assignments	
	XCERx	Block Assigned	Bit in XCERx	Channel Assigned
	XCERE	Block 4	XCE0	Channel 64
			XCE1	Channel 65
			XCE2	Channel 66
			:	:
			XCE15	Channel 79
	XCERF	Block 5	XCE0	Channel 80
			XCE1	Channel 81
			XCE2	Channel 82
			:	:
			XCE15	Channel 95
	XCERG	Block 6	XCE0	Channel 96
			XCE1	Channel 97
			XCE2	Channel 98
			:	:
			XCE15	Channel 111
	XCERH	Block 7	XCE0	Channel 112
			XCE1	Channel 113
			XCE2	Channel 114
			:	:
			XCE15	Channel 127

Controller Area Network (CAN)

This chapter contains a general description of the Controller Area Network (CAN) module. The CAN module is a serial communications protocol which efficiently supports distributed real-time control with a high level of security. The CAN module supports bit-rates up to 1 Mbit/s and is compliant with the ISO11898-1 (CAN 2.0B) protocol specification.

Topic	Page
21.1 Overview.....	2102
21.2 Configuring Device Pins	2104
21.3 Operating Modes	2104
21.4 Multiple Clock Source	2108
21.5 Interrupt Functionality	2108
21.6 Global Power-down Mode	2109
21.7 Local Power-down Mode	2109
21.8 Parity Check Mechanism	2110
21.9 Debug Mode	2111
21.10 Module Initialization	2111
21.11 Configuration of Message Objects	2111
21.12 Message Handling	2113
21.13 CAN Bit Timing	2118
21.14 Message Interface Register Sets	2126
21.15 Message RAM	2128
21.16 Registers	2133

21.1 Overview

This device uses the popular CAN IP known as D_CAN.

21.1.1 Features

The CAN implements the following features:

- CAN protocol version ISO11898-1 (CAN 2.0B)
- Bit rates up to 1 MBit/s
- Multiple clock sources
- 32 message objects
- 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
- Global power down and wake-up support

21.1.2 Functional Description

The CAN performs CAN protocol communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 A, B). The bit rate can be programmed to values up to 1 MBit/s. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

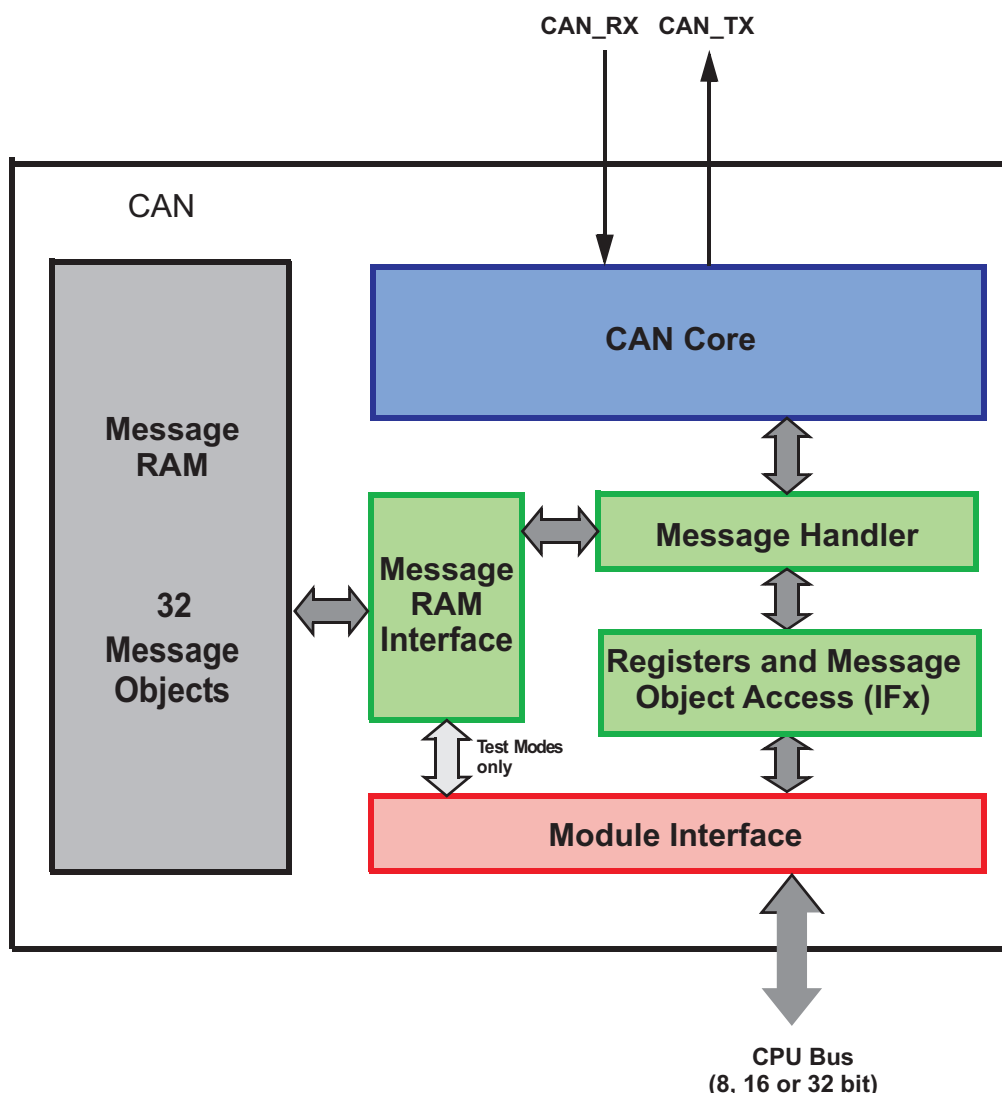
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests.

The register set of the CAN may be accessed directly by the CPU via the module interface. These registers are used to control/configure the CAN core and the message handler, and to access the message RAM.

21.1.3 Block Diagram

Figure 21-1. CAN Block Diagram



21.1.3.1 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.

21.1.3.2 Message Handler

The message handler is a state machine which controls the data transfer between the single ported 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.

21.1.3.3 Message RAM

The CAN message RAM enables the storage of 32 CAN messages.

21.1.3.4 Registers and Message Object Access (IFx)

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU 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 21.14](#). 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.

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.

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.

21.3 Operating Modes

21.3.1 Software Initialization

The software initialization mode is entered by setting the **Init** bit in the CAN Control register, either by software, 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 set up 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.

Resetting 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 21.13](#)).

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.

NOTE: The CAN module uses a special addressing scheme to support byte accesses. This is the same addressing that is used on the USB module, which is described in [Section 22.3.4](#). For ease of use, it is recommended to only make 32-bit accesses to the CAN registers. However, at higher optimization levels, the compiler may split a 32-bit access into two sequential 16-bit accesses, which will corrupt the register value. A compiler fix is in development. In the meantime, 16-bit accesses can be used as a workaround. 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.

21.3.2 CAN Message Transfer (Normal Operation)

Once the CAN is initialized and **Init** bit is reset to zero, the CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored into their appropriate message objects if they pass acceptance filtering. The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence, when, for example, the identifier mask is used, the arbitration 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 (arbitration and control bits set up during configuration and leaving unchanged 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.

21.3.2.1 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 bit DAR in the CAN Control register. Further details to this mode are provided in [Section 21.12.3](#).

21.3.2.2 Auto-Bus-On

Per default, after the CAN has entered bus-off state, the CPU can start a bus-off-recovery sequence by resetting 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 bit ABO. 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 goes Bus-Off due to massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit has been reset by the CPU 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. At the end of the bus-off recovery sequence, the error counters will be reset.

21.3.3 Test Modes

The CAN provides several test modes which are mainly intended for self test.

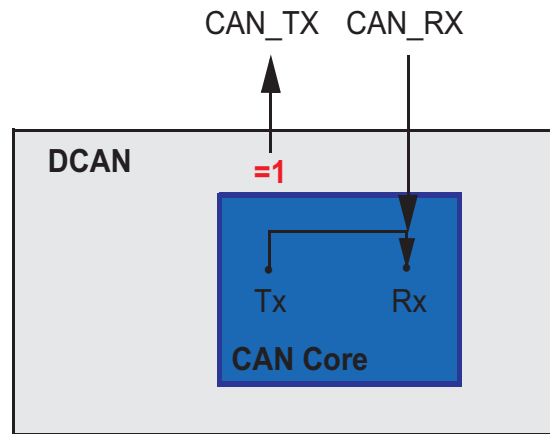
For all test modes, bit **Test** in the CAN Control register needs to be set to one. This enables write access to the Test register.

21.3.3.1 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, these are internally routed to the CAN Core.

Figure 21-2 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 to one. In ISO 11898-1, the silent mode is called the bus monitoring mode.

Figure 21-2. CAN Core in Silent Mode



21.3.3.2 Loopback Mode

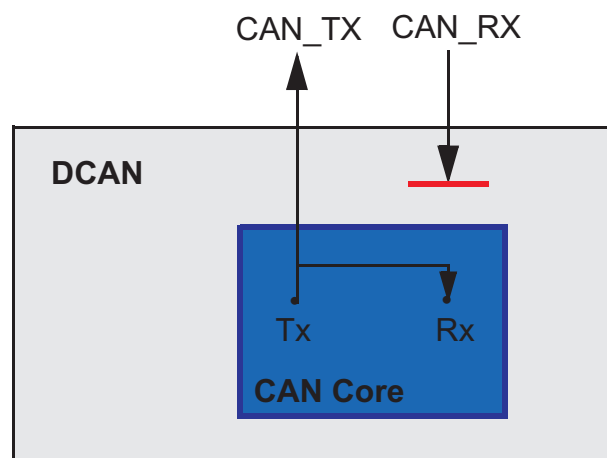
The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the 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 21-3 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 TTST register to one.

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 21.3.3.3](#).

Figure 21-3. CAN Core in Loopback Mode



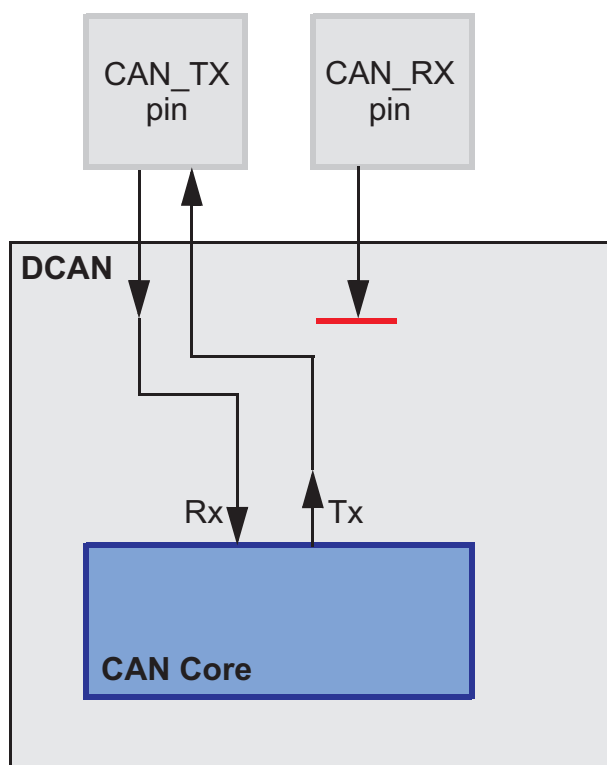
21.3.3.3 External Loopback Mode

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 input of 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 bit **ExL** in Test Register to one.

Figure 21-4 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.

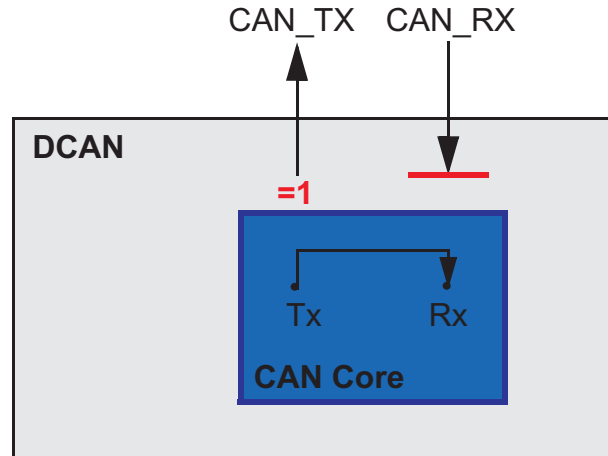
Figure 21-4. CAN Core in External Loopback Mode



21.3.3.4 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. This mode can be used for a "Hot Selftest," that is, 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 21-5 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 21-5. CAN Core in Loopback Combined with Silent Mode


21.4 Multiple Clock Source

Three clock domains are provided to the CAN module for generating the CAN bit timing: the external oscillator clock (X1/X2), the system clock, and the GPIO AUXCLKIN.

The system module reference guide 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 MBaud an oscillator frequency of 8 MHz or higher has to be used.

21.5 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 WakeUpPnd, 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 resp. 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 that is the source of the interrupt, may 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, 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.

21.5.1 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 21.15.1](#). Message object interrupts can be routed to either CAN0INT or CAN1INT line, controlled by the Interrupt Multiplexer register.

21.5.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status register belong to the status change interrupts. The status change interrupt group can be enabled by bit SIE in 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 Control Register.

NOTE: Reading the Error and Status register will clear the WakeUpPnd flag. If in global power down mode, the WakeUpPnd flag is cleared by such a read access before the CAN module has been waken up by the system, the CAN may re-assert the WakeUpPnd flag, and a second interrupt may occur. (see also [Section 21.6.2](#)).

21.5.3 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 this register.

21.6 Global Power-down Mode

The Delfino™ architecture supports a centralized global power-down control over the peripheral modules through the sleep and deep sleep modes present in the system control peripheral.

21.6.1 Entering Global Power-down Mode

The global power down mode for the CAN is requested by configuring the appropriate bits in the sleep mode clock gating control(SCGC) or deep sleep mode clock gating Control(DCGC) register and enabled either of these modes (sleep or deep sleep).

The CAN then finishes all transmit requests of the message objects. When all requests are done, the CAN waits until a bus idle state is recognized. Then it will automatically set the Init bit to indicate that the global power-down mode has been entered.

21.6.2 Wakeup from Global Power-down Mode

If the CAN module is in global power-down mode, a CAN bus activity detection circuit is active. On occurrence of a dominant CAN bus level, the CAN will set the WakeUpPnd bit in Error and Status register.

If status interrupts are enabled, also an interrupt will be generated. This interrupt could be used by the application to wakeup the CAN. For this, the application needs to clear the Init bit in CAN Control register.

After the Init bit has been cleared, the CAN module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes bus-active again.

NOTE: The CAN transceiver circuit has to stay active during CAN bus activity detection. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down mode is lost.

21.7 Local Power-down Mode

Besides the centralized power-down mechanism controlled by the system control module, the CAN supports a local power-down mode which can be controlled within the DCAN control registers.

21.7.1 Entering Local Power-down Mode

The local power-down mode is requested by setting the PDR bit in CAN Control register.

The CAN then finishes all transmit requests of the message objects. When all requests are done, CAN waits until a bus idle state is recognized. Then it will automatically set the Init bit in the CAN Control register to prevent any further CAN transfers, and it will also set the PDA bit in CAN Error and Status register. With setting the PDA bits, the CAN module indicates that the local power down mode has been entered.

During local power-down mode, the internal clocks of the CAN module are turned off, but there is a wake up logic which can be active, if enabled. Also the actual contents of the control registers can be read back.

21.7.2 Wakeup from Local Power-down Mode

There are two ways to wake up the CAN from local power down mode:

1. The application could wake up the CAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control register
2. A CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control register.

If this circuit is active, on occurrence of a dominant CAN bus level, the CAN will automatically start the wake up sequence. It will clear the PDR bit in CAN Control register and also clear the PDA bit in Error and Status register. The WakeUpPnd bit in CAN Error and Status register will be set. If Status Interrupts are enabled, also an interrupt will be generated. Finally the Init bit in CAN control register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes bus-active again.

NOTE: In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the init bit, these messages may be sent. In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the init bit, these messages may be sent.

21.8 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 PMD bit field in CAN Control register. In case of 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 definition is equivalent to: the parity bit will be set, if the number of 1 bits in the data is odd.

21.8.1 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 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.

21.9 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 either wait until a started transmission or reception will be finished and Bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit IDS. 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 21.15.3](#)).

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 DMAActive flag by r/w)

21.10 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 21.11](#).

For the configuration of the Bit Timing, see [Section 21.13.2](#).

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.

21.11 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.

21.11.1 Configuration of a Transmit Object for Data Frames

Figure 21-6 shows how a transmit object can be initialized.

Figure 21-6. 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 21.12.8](#). Identifier masking must be disabled (UMask = '0') if no remote frames are allowed to set the TxRqst bit (RmtEn = '0').

21.11.2 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.

21.11.3 Configuration of a Single Receive Object for Data Frames

Figure 21-7 shows how a receive object for data frames can be initialized.

Figure 21-7. 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'.
- The data length code (DLC[3:0]) is given by the application. When the message handler stores a data frame in the message object, it will store the received data length code and eight 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.

21.11.4 Configuration of a Single Receive Object for Remote Frames

Figure 21-8 shows how a receive object for remote frames can be initialized.

Figure 21-8. 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 21.11.5](#).
- 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 21.12.8](#).
- 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.

21.11.5 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 have to 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.

21.12 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 has to 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 is accepted for the same message object will be overwritten. Messages may be read interrupt-driven or after polling of NewDat.

21.12.1 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.

21.12.2 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.

21.12.3 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.

21.12.4 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 21.12.3](#).

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

21.12.5 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 only update the data bytes of a message to be transmitted, bits [23:16] of the Command register should be set 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.

21.12.6 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 of 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.

21.12.7 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 (supposedly 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.

21.12.8 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

21.12.9 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.

21.12.10 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.

21.12.11 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.

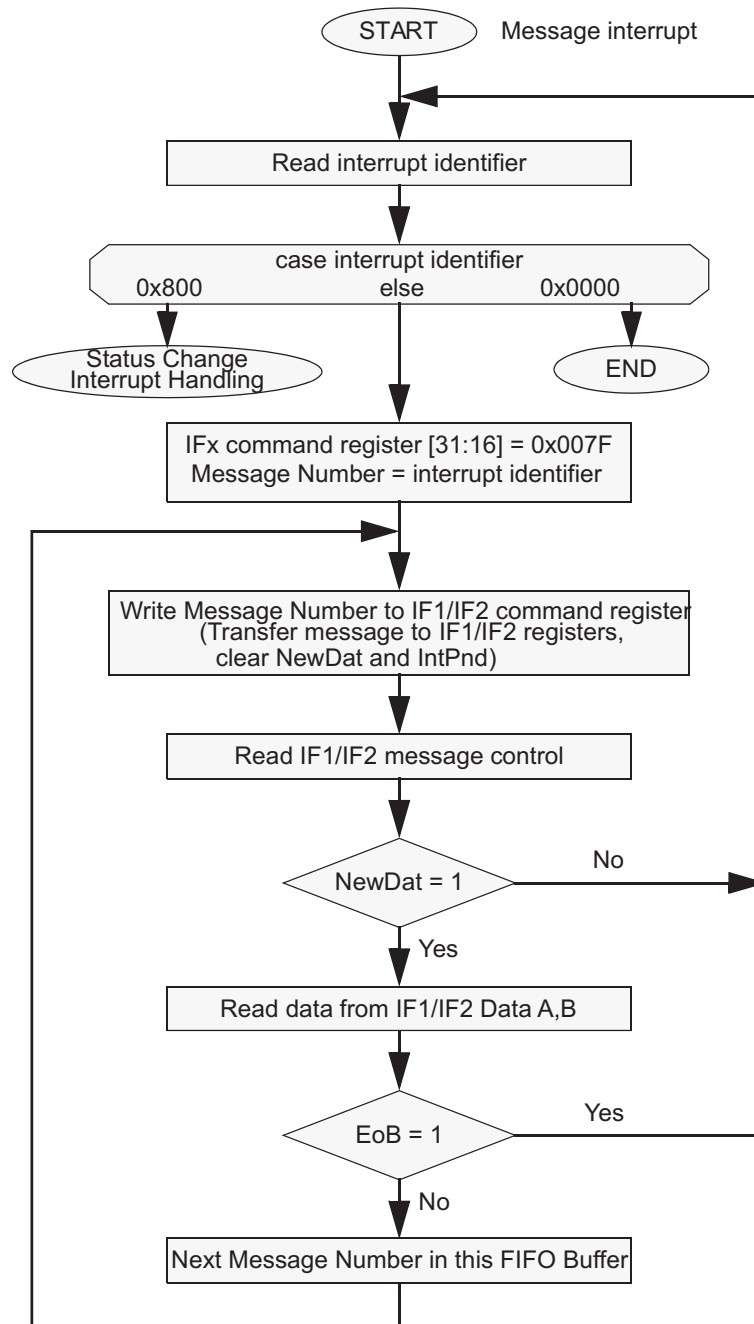
Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to '0', all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = '1') and therefore overwrite previous messages in this message object.

21.12.12 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 21-9](#).

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 can not 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 21-9. CPU Handling of a FIFO Buffer (Interrupt Driven)


21.13 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.

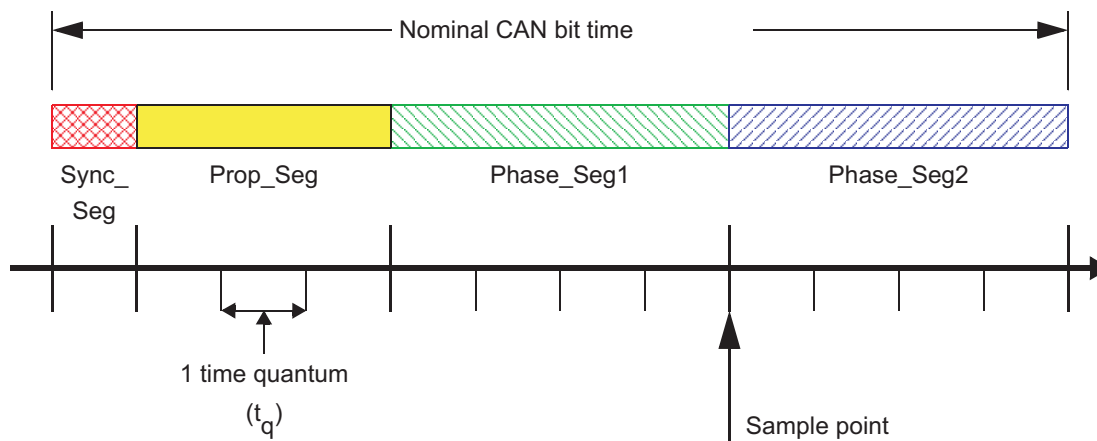
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

21.13.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 21-10):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 21-10. Bit Timing



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:

$$t_q = \text{Baud Rate Prescaler} / \text{CAN_CLK}$$

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 21-1 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

Table 21-1. 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.

21.13.1.1 Synchronization Segment

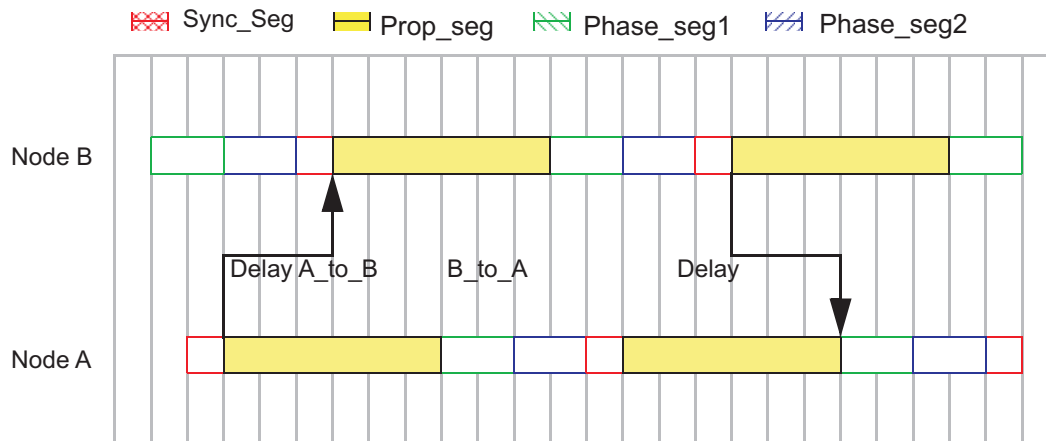
The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

21.13.1.2 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 21-11](#) shows the phase shift and propagation times between two CAN nodes.

Figure 21-11. The Propagation Time Segment



$$\text{Delay A_to_B} \geq \text{node output delay(A)} + \text{bus line delay(A/B)} + \text{node input delay(B)}$$

$$\text{Prop_Seg} \geq \text{Delay A_to_B} + \text{Delay B_to_A}$$

$$\text{Prop_Seg} \geq 2 \cdot [\max(\text{node output delay} + \text{bus line delay} + \text{node input delay})]$$

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 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.

21.13.1.3 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.

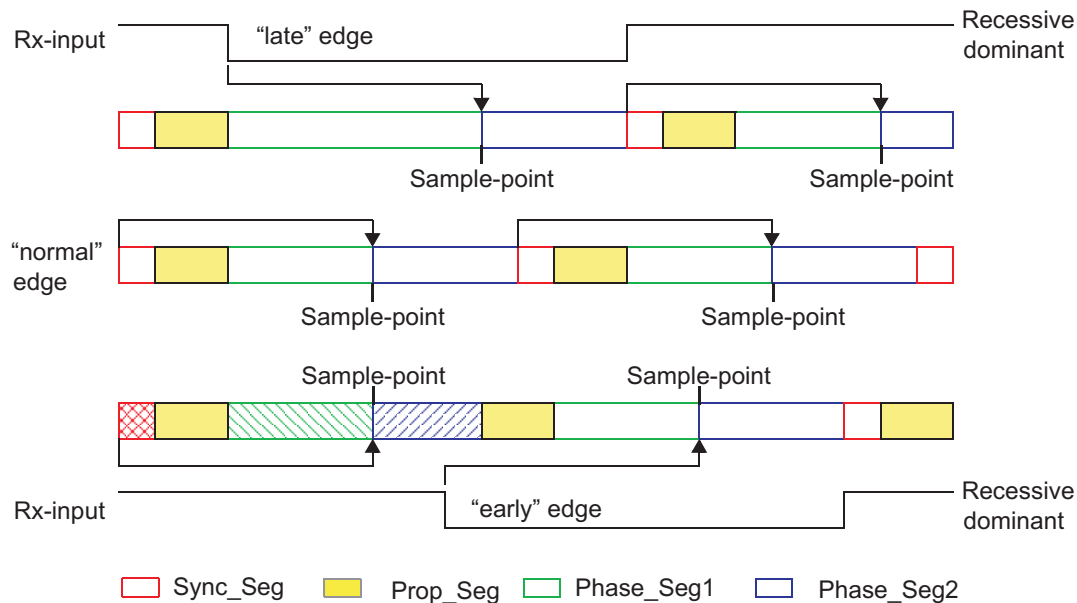
If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and 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 21-12](#) 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 21-12. 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.

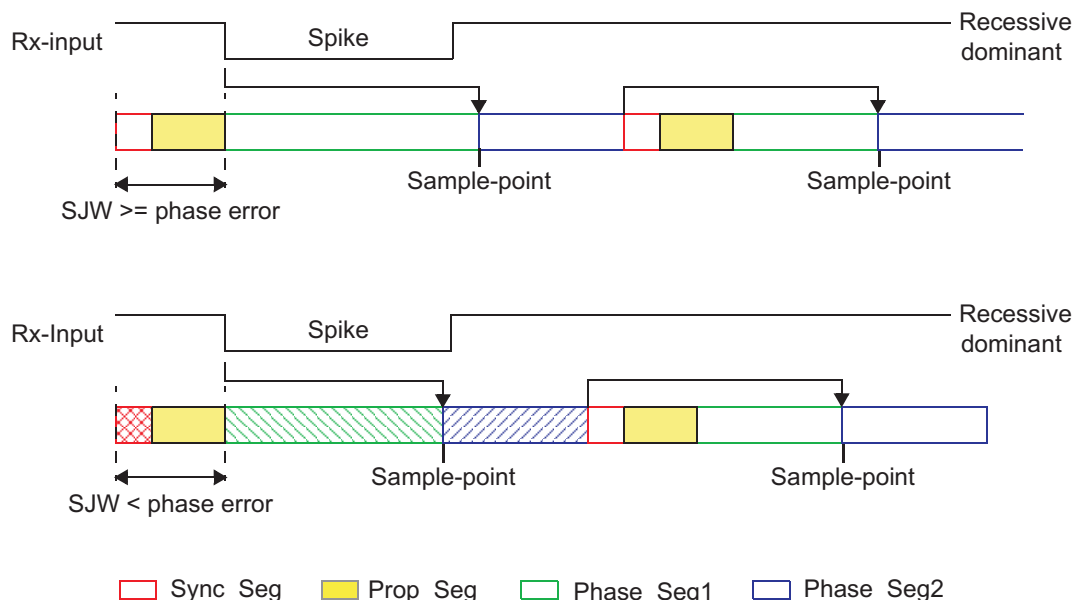
The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

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 21-13](#) 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 21-13. Filtering of Short Dominant Spikes



21.13.1.4 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):

(8)

$$I: df \leq \frac{\min(TSeg1, Tseg2)}{2 \times (13 \times (bit_time - TSeg2))}$$

(9)

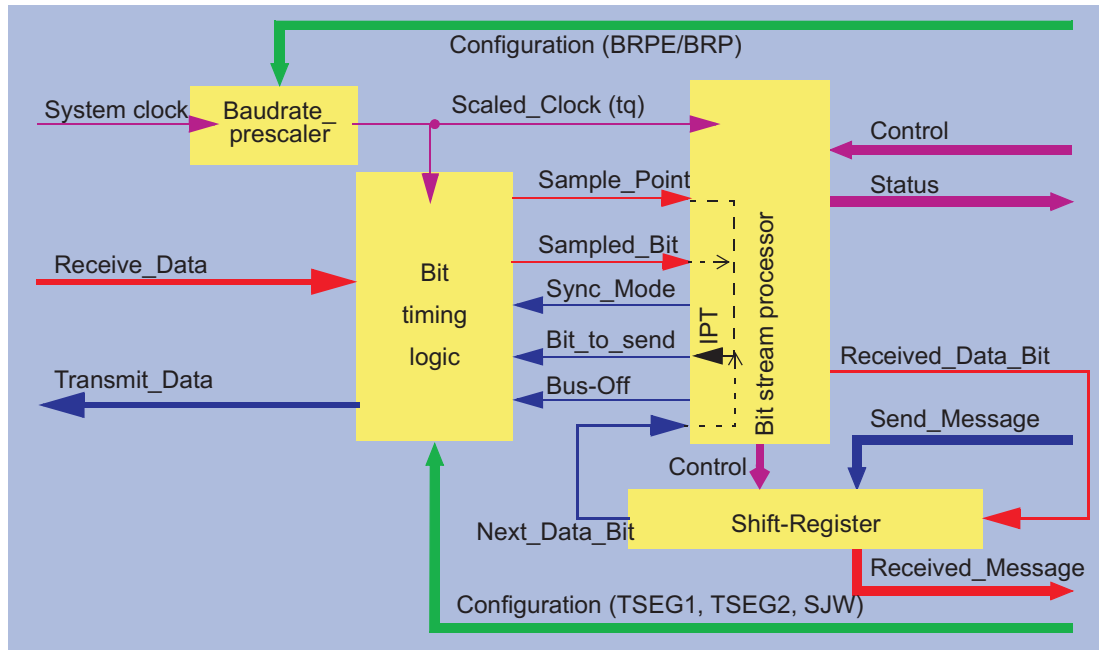
$$II: df \leq \frac{SJW}{20 \times bit_time}$$

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 μs) with a bus length of 40 m.

21.13.2 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 21-14).

Figure 21-14. 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 IPC 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.

21.13.2.1 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 21.13.1.4](#).

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:

$(\text{Phase_Seg2} - 1) \& (\text{Phase_Seg1} + \text{Prop_Seg} - 1) \&$

$(\text{SynchronizationJumpWidth} - 1) \& (\text{Prescaler} - 1)$

21.13.2.2 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_{Prop}	700 ns	=	$2 \cdot \text{delays} = 7 \cdot t_q$
t_{SJW}	100 ns	=	$1 \cdot t_q$
t_{TSeg1}	800 ns	=	$t_{\text{Prop}} + t_{\text{SJW}}$
t_{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	0.43 %	=	$\frac{\min(t_{\text{TSeg1}}, t_{\text{TSeg2}})}{2 \times (13 \times (\text{bit_time} - t_{\text{TSeg2}}))}$

$$= \frac{0,1\mu\text{s}}{2 \times 13 \times (1\mu\text{s} - 0,1\mu\text{s}))}$$

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.

21.13.2.3 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 μs	=	$2 \cdot t_{\text{CAN_CLK}}$
delay of bus driver	200 ns	=	
delay of receiver circuit	80 ns	=	
delay of bus line (40m)	220 ns	=	
t_{Prop}	1 μs	=	$1 \cdot t_q$
t_{SJW}	4 μs	=	$4 \cdot t_q$
t_{TSeg1}	5 μs	=	$t_{\text{Prop}} + t_{\text{SJW}}$
t_{TSeg2}	4 μs	=	Information Processing Time + $4 \cdot t_q$
$t_{\text{Sync-Seg}}$	1 μs	=	$1 \cdot t_q$
bit time	10 μs	=	$t_{\text{Sync-Seg}} + t_{\text{TSeg1}} + t_{\text{TSeg2}}$
tolerance for CAN_CLK	3.08 %	=	$\frac{\min(\text{TSeg1}, \text{TSeg2})}{2 \times (13 \times (\text{bit_time} - \text{TSeg2}))}$
		=	$\frac{4\mu\text{s}}{2 \times (13 \times (9\mu\text{s} - 4\mu\text{s}))}$

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.

21.14 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 / 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 21.15.1](#)) 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.

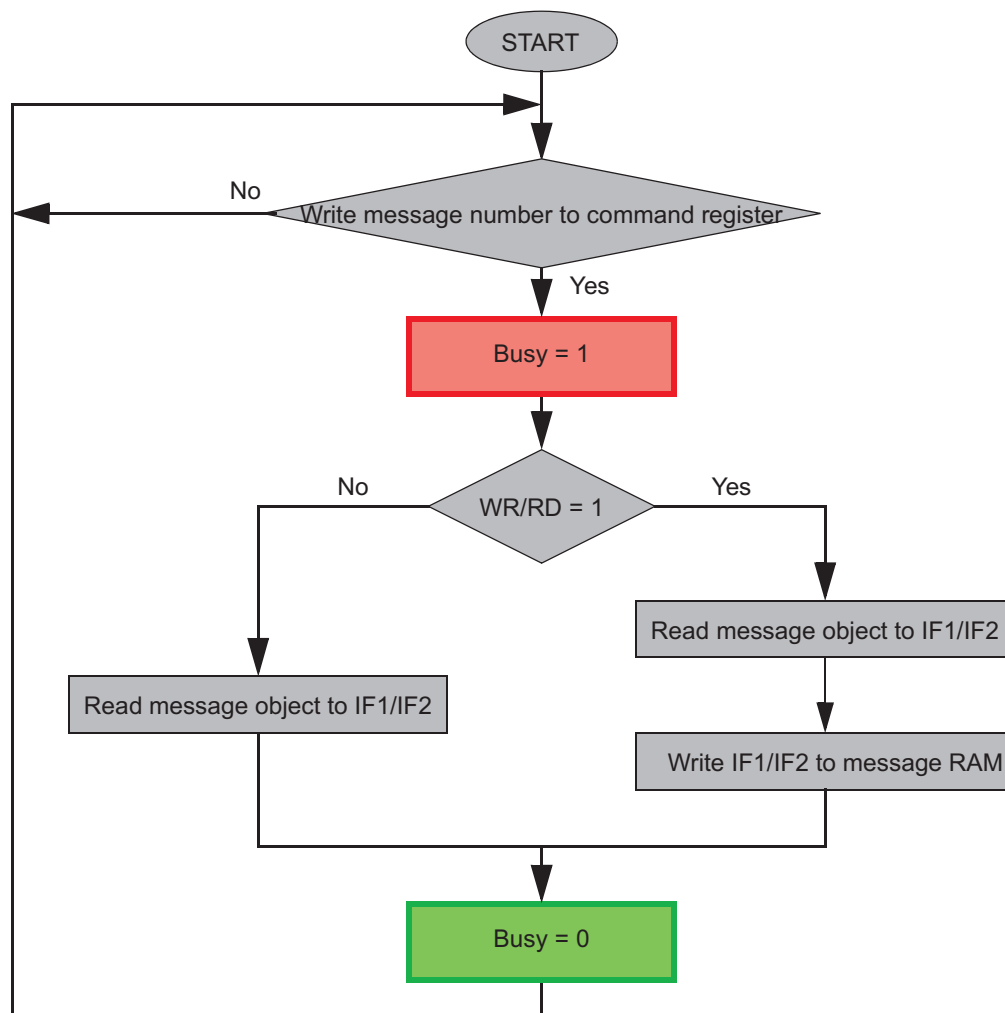
NOTE: Only 32-bit accesses to the CAN registers are allowed, as 16-bit accesses may not work as expected.

21.14.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 registers Sets control the data transfer to and from the message object. The Command Register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command 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 21-15](#)).

Figure 21-15. Data Transfer Between IF1 / IF2 Registers and Message RAM



21.14.2 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 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, 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 CAN internal IF3 update is complete, an IF3 interrupt can also be generated.

NOTE: The IF3 register set can not be used for transferring data into message objects.

21.15 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.

21.15.1 Structure of Message Objects

Figure 21-16 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 21-16. 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 21-2. Message Object Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	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.		

Table 21-2. Message Object Field Descriptions (continued)

Name	Value	Description
UMask	0 1	Use Acceptance Mask Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering. 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] ID[28:18]	Message Identifier 29-bit ("extended") identifier bits 11-bit ("standard") identifier bits
Msk[28:0]	0 1	Identifier Mask The corresponding bit in the message identifier is not used for acceptance filtering (don't care). The corresponding bit in the message identifier is used for acceptance filtering.
Xtd	0 1	Extended Identifier The 11-bit ("standard") identifier will be used for this message object. The 29-bit ("extended") identifier will be used for this message object.
MXtd	0 1	Mask Extended Identifier The extended identifier bit (IDE) has no effect on the acceptance filtering. 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 1	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. 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 = one).
MDir	0 1	Mask Message Direction The message direction bit (Dir) has no effect on the acceptance filtering. The message direction bit (Dir) is used for acceptance filtering.
EOB	0 1	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. 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 1	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. The message handler or the CPU has written new data into the data bytes of this message object.
MsgLst	0 1	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. 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 1	Receive Interrupt Enable IntPnd will not be triggered after the successful reception of a frame. IntPnd will be triggered after the successful reception of a frame.
TxIE	0 1	Transmit Interrupt Enable IntPnd will not be triggered after the successful transmission of a frame. IntPnd will be triggered after the successful transmission of a frame.

Table 21-2. Message Object Field Descriptions (continued)

Name	Value	Description
IntPnd	0	Interrupt Pending
	1	This message object is not the source of an interrupt. 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
	1	At the reception of a remote frame, TxRqst is not changed. At the reception of a remote frame, TxRqst is set. Note: See Section 21.12.8 for details on the setup of RmtEn and UMask for remote frames.
TxRqst	0	Transmit Request
	1	This message object is not waiting for a transmission. The transmission of this message object is requested and is not yet done.
DLC[3:0]	0	Data length code
	1	Data frame has 0-8 data bits. 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		1st data byte of a CAN data frame
Data 1		2nd data byte of a CAN data frame
Data 2		3rd data byte of a CAN data frame
Data 3		4th data byte of a CAN data frame
Data 4		5th data byte of a CAN data frame
Data 5		6th data byte of a CAN data frame
Data 6		7th data byte of a CAN data frame
Data 7		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.

21.15.2 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, etc.

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 21-3. Message RAM Addressing in Debug Mode

Message Object Number	Offset From Base Address	Word Number	Debug Mode ⁽¹⁾
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
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

⁽¹⁾ See [Section 21.15.3](#).

21.15.3 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 21-17. Message RAM Representation in Debug Mode

31/ 15	30/ 14	29/ 13	28/ 12	27/ 11	26/ 10	25/ 9	24/ 8	23/ 7	22/ 6	21/ 5	20/ 4	19/ 3	18/ 2	17/ 1	16/ 0
MsgAddr + 0x00															
Reserved															
Reserved												Parity[4:0]			
MsgAddr + 0x04															
MXtd	MDir	Rsvd	Msk[28:16]												
Msk[15:0]															
MsgAddr + 0x08															
Rsvd	Xtd	Dir	ID[28:16]												
ID[15:0]															
MsgAddr + 0x0C															
Reserved															
Rsvd	MsgLs t	Rsvd	UMask	TxIE	RxIE	RmtEn	Rsvd	EOB	Reserved			DLC[3:0]			
MsgAddr + 0x10															
Data 3								Data 2							
Data 1								Data 0							
MsgAddr + 0x14															
Data 7								Data 6							
Data 5								Data 4							

21.16 Registers

21.16.1 CAN Base Addresses

Table 21-4. CAN Base Addresses Table

Device Registers	Register Name	Start Address	End Address
CanaRegs	CAN_REGS	0x0004_8000	0x0004_87FF
CanbRegs	CAN_REGS	0x0004_A000	0x0004_A7FF

21.16.2 CAN_REGS Registers

Table 21-5 lists the memory-mapped registers for the CAN_REGS. All register offset addresses not listed in Table 21-5 should be considered as reserved locations and the register contents should not be modified.

Table 21-5. CAN_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	CAN_CTL	CAN Control Register	EALLOW	Go
4h	CAN_ES	Error and Status Register		Go
8h	CAN_ERRRC	Error Counter Register		Go
Ch	CAN_BTR	Bit Timing Register		Go
10h	CAN_INT	Interrupt Register		Go
14h	CAN_TEST	Test Register		Go
1Ch	CAN_PERR	CAN Parity Error Code Register		Go
20h	CAN_REL	CAN Core Release Register		Go
40h	CAN_RAM_INIT	CAN RAM Initialization Register		Go
50h	CAN_GLB_INT_EN	CAN Global Interrupt Enable Register		Go
54h	CAN_GLB_INT_FLG	CAN Global Interrupt Flag Register		Go
58h	CAN_GLB_INT_CLR	CAN Global Interrupt Clear Register		Go
80h	CAN_ABOTR	Auto-Bus-On Time Register		Go
84h	CAN_TXRQ_X	CAN Transmission Request X Register		Go
88h	CAN_TXRQ_21	CAN Transmission Request 2_1 Register		Go
98h	CAN_NDAT_X	CAN New Data X Register		Go
9Ch	CAN_NDAT_21	CAN New Data 2_1 Register		Go
ACh	CAN_IPEN_X	CAN Interrupt Pending X Register		Go
B0h	CAN_IPEN_21	CAN Interrupt Pending 2_1 Register		Go
C0h	CAN_MVAL_X	CAN Message Valid X Register		Go
C4h	CAN_MVAL_21	CAN Message Valid 2_1 Register		Go
D8h	CAN_IP_MUX21	CAN Interrupt Multiplexer 2_1 Register		Go
100h	CAN_IF1CMD	IF1 Command Register		Go
104h	CAN_IF1MSK	IF1 Mask Register		Go
108h	CAN_IF1ARB	IF1 Arbitration Register		Go
10Ch	CAN_IF1MCTL	IF1 Message Control Register		Go
110h	CAN_IF1DATA	IF1 Data A Register		Go
114h	CAN_IF1DATB	IF1 Data B Register		Go
120h	CAN_IF2CMD	IF2 Command Register		Go
124h	CAN_IF2MSK	IF2 Mask Register		Go
128h	CAN_IF2ARB	IF2 Arbitration Register		Go
12Ch	CAN_IF2MCTL	IF2 Message Control Register		Go
130h	CAN_IF2DATA	IF2 Data A Register		Go
134h	CAN_IF2DATB	IF2 Data B Register		Go
140h	CAN_IF3OBS	IF3 Observation Register		Go
144h	CAN_IF3MSK	IF3 Mask Register		Go
148h	CAN_IF3ARB	IF3 Arbitration Register		Go
14Ch	CAN_IF3MCTL	IF3 Message Control Register		Go
150h	CAN_IF3DATA	IF3 Data A Register		Go
154h	CAN_IF3DATB	IF3 Data B Register		Go
160h	CAN_IF3UPD	IF3 Update Enable Register		Go

21.16.2.1 CAN_CTL Register (Offset = 0h) [reset = 1401h]

CAN_CTL is shown in [Figure 21-18](#) and described in [Table 21-6](#).

CAN Control Register

Figure 21-18. CAN_CTL Register

31	30	29	28	27	26	25	24
RESERVED						WUBA	PDR
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			RESERVED	RESERVED	RESERVED	IE1	INITDBG
R-0h			R-0h	R-0h	R-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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-6. CAN_CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	WUBA	R/W	0h	Automatic wake up on bus activity Enable Bit: This bit is used to enable/disable Automatic wake up on bus activity, when in local power down mode. 0 No detection of a dominant CAN bus level while in local power down mode. 1 Detection of a dominant CAN bus level while in local power down mode is enabled. On occurrence of a dominant CAN bus level, the wake up
24	PDR	R/W	0h	Power Down Mode Request Bit: This bit is used to put the CAN module in local power down mode. 0 No application request for local low power down mode. If the application has cleared this bit while CAN is in power down mode, the INIT bit has to be cleared as well. 1 Power down mode has been requested by application. The CAN module will acknowledge this mode by setting the PDA bit in Error and Status Register. The local clocks will be turned off by CAN internal logic.
23-21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17	IE1	R/W	0h	Interrupt line 1 Enable Bit: This bit is used to enable/disable interrupt line 1 (CANN_INT1) 0 Disabled. 1 Enabled.
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 is ready for debug accesses.

Table 21-6. CAN_CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
14	RESERVED	R	0h	Reserved
13-10	PMD	R/W	5h	Parity on/off - 0101 Parity function disabled xxxx Parity function enabled
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
8	IDS	R/W	0h	Interruption Debug Support Enable - 0 When Debug mode is requested, CAN will wait for a started transmission or reception to be completed before entering Debug mode 1 When Debug mode is requested, CAN will interrupt any transmission or reception, and enter Debug mode immediately.
7	Test	R/W	0h	Test Mode Enable - Normal Operation Test Mode
6	CCE	R/W	0h	Configuration Change Enable - 0 The CPU has no write access to the configuration registers. 1 The CPU has write access to the configuration registers (when Init bit is set).
5	DAR	R/W	0h	Disable Automatic Retransmission - 0 Automatic Retransmission of not successful messages enabled. 1 Automatic Retransmission disabled.
4	RESERVED	R	0h	Reserved
3	EIE	R/W	0h	Error Interrupt Enable Disabled - 0 PER, BOff and EWarn bits can not generate an interrupt. 1 Enabled- PER, BOff and EWarn bits can generate an interrupt at CAN0INT line and affect the Interrupt Register.
2	SIE	R/W	0h	Status Change Interrupt Enable Disabled - 0 WakeUpPnd, RxOk, TxOk and LEC bits can not generate an interrupt. 1 Enabled - WakeUpPnd, RxOk, TxOk and LEC can generate an interrupt at CAN0INT line and affect the Interrupt Register.
1	IE0	R/W	0h	Interrupt line 0 Enable Disabled - 0 Module Interrupt CAN0INT is always low. 1 Enabled - Interrupts will assert line CAN0INT to one line remains active until pending interrupts are processed.
0	Init	R/W	1h	Initialization Normal Operation Initialization mode is entered

21.16.2.2 CAN_ES Register (Offset = 4h) [reset = 3h]

CAN_ES is shown in [Figure 21-19](#) and described in [Table 21-7](#).

Error and Status Register

Figure 21-19. 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					PDA	WakeUpPnd	PER
R-0h					R-0h	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-3h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-7. CAN_ES Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10	PDA	R	0h	Local power down mode acknowledge: 0 CAN is not in local power down mode. 1 Application request for setting CAN to local power down mode was successful. CAN is in local power down mode.
9	WakeUpPnd	R	0h	Wakeup pending: This bit can be used by the CPU to identify the CAN as the source to wake up the system. The bit will be reset after the CPU reads the register. 0 No Wake Up is requested by CAN. 1 CAN has initiated a wake up of the system due to dominant CAN bus while module power down.
8	PER	R	0h	Parity/double bit Error Detected: This bit will be reset after the CPU reads the register. 0 No parity/double bit error has been detected since last read access. 1 The parity check/SECEDED mechanism has detected a parity/double bit error in the Message RAM.
7	BOff	R	0h	Bus-off Status Bit: 0 The CAN module is not Bus-Off state. 1 The CAN module is in Bus-Off state.
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.
5	EPass	R	0h	Error Passive State - 0 On CAN Bus error, the CAN could send active error frames. 1 The CAN Core is in the error passive state as defined in the CAN Specification.

Table 21-7. CAN_ES Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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.</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>
2-0	LEC	R	3h	<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>

21.16.2.3 CAN_ERRC Register (Offset = 8h) [reset = 0h]

CAN_ERRC is shown in [Figure 21-20](#) and described in [Table 21-8](#).

Error Counter Register

Figure 21-20. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-8. 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.
14-8	REC	R	0h	Receive Error Counter: Actual state of the Receive Error Counter. (values from 0 to 127).
7-0	TEC	R	0h	Transmit Error Counter - Actual state of the Transmit Error Counter. (values from 0 to 255).

21.16.2.4 CAN_BTR Register (Offset = Ch) [reset = 2301h]

CAN_BTR is shown in [Figure 21-21](#) and described in [Table 21-9](#).

Bit Timing Register

Figure 21-21. 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					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-9. 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.
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.
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.
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.
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.

21.16.2.5 CAN_INT Register (Offset = 10h) [reset = 0h]

CAN_INT is shown in [Figure 21-22](#) and described in [Table 21-10](#).

Interrupt Register

Figure 21-22. 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															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-10. CAN_INT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	INT1ID	R	0h	<p>Interrupt Identifier 1: The number here indicates the source of the interrupt.</p> <p>0x00 No interrupt is pending.</p> <p>0x01-0x20 Number of mailbox (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>The CANn_INT1 interrupt line remains active until INT0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared.</p> <p>A message interrupt is cleared by clearing the mailbox's IntPnd bit.</p> <p>Among the message interrupts, the mailbox's interrupt priority decreases with increasing message number.</p>
15-0	INT0ID	R	0h	<p>Interrupt Identifier (the number here indicates the source of the interrupt)</p> <p>0x0000 No interrupt is pending</p> <p>0x0001- Number of message object which caused the interrupt.</p> <p>0x0080 0x0081- Unused</p> <p>0x7FFF 0x8000 Error and Status Register value is not 0x07.</p> <p>0x8001- Unused If several interrupts are pending, the CAN Interrupt Register will point to the pending 0xFFFF interrupt with the highest priority.</p> <p>The CAN0INT 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>

21.16.2.6 CAN_TEST Register (Offset = 14h) [reset = 0h]

CAN_TEST is shown in [Figure 21-23](#) and described in [Table 21-11](#).

Test Register

Figure 21-23. 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		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-11. 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.
8	EXL	R/W	0h	External Loop Back Mode: 0 Disabled. 1 Enabled.
7	RX	R	0h	Monitors the actual value of the CANRX pin: 0 The CAN bus is dominant. 1 The CAN bus is recessive.
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.
4	LBACK	R/W	0h	Loop Back Mode: 0 Disabled. 1 Enabled.
3	SILENT	R/W	0h	Silent Mode: 0 Disabled. 1 Enabled.
2-0	RESERVED	R	0h	Reserved

21.16.2.7 CAN_PERR Register (Offset = 1Ch) [reset = 100h]

CAN_PERR is shown in [Figure 21-24](#) and described in [Table 21-12](#).

CAN Parity Error Code Register

Figure 21-24. 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					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-12. 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, see section 1.18.5). For the SECEDED implementation the Word Number is 'Rsvd' and it will read always as 0x0.
7-0	MSG_NUM	R	0h	0x01-0x21 Mailbox number where parity/double bit error has been detected

21.16.2.8 CAN_REL Register (Offset = 20h) [reset = A3170504h]

CAN_REL is shown in [Figure 21-25](#) and described in [Table 21-13](#).

CAN Core Release Register

Figure 21-25. CAN_REL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL				STEP				SUBSTEP				YEAR			
R-Ah				R-3h				R-1h				R-7h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON								DAY							
R-5h								R-4h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-13. CAN_REL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	REL	R	Ah	Core Release One digit, BCD-coded.
27-24	STEP	R	3h	Step of Core Release One digit, BCD-coded.
23-20	SUBSTEP	R	1h	Substep of Core Release One digit, BCD-coded.
19-16	YEAR	R	7h	Design Time Stamp, Year One-digit BCD-coded year value from the CAN synthesis time stamp. This constant value is for TI internal use only.
15-8	MON	R	5h	Design Time Stamp, Month Two-digit BCD-coded month value from the CAN synthesis time stamp. This constant value is for TI internal use only.
7-0	DAY	R	4h	Design Time Stamp, Day Two-digit BCD-coded day value from the CAN synthesis time stamp. This constant value is for TI internal use only.

21.16.2.9 CAN_RAM_INIT Register (Offset = 40h) [reset = 5h]

CAN_RAM_INIT is shown in [Figure 21-26](#) and described in [Table 21-14](#).

CAN RAM Initialization Register

Figure 21-26. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-14. 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
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.
3	KEY3	R/W	0h	See Key 0
2	KEY2	R/W	1h	See Key 0
1	KEY1	R/W	0h	See Key 0
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.

21.16.2.10 CAN_GLB_INT_EN Register (Offset = 50h) [reset = 0h]

CAN_GLB_INT_EN is shown in [Figure 21-27](#) and described in [Table 21-15](#).

CAN Global Interrupt Enable Register

Figure 21-27. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-15. 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 CAN INT1 0 = CAN INT1 does not generate interrupt to PIE 1= CAN INT1 generates interrupt to PIE if interrupt condition occurs
0	GLBINT0_EN	R/W	0h	Global Interrupt Enable for CAN INT0 0 = CAN INT0does not generate interrupt to PIE 1= CAN INT0 generates interrupt to PIE if interrupt condition occurs

21.16.2.11 CAN_GLB_INT_FLG Register (Offset = 54h) [reset = 0h]

CAN_GLB_INT_FLG is shown in [Figure 21-28](#) and described in [Table 21-16](#).

CAN Global Interrupt Flag Register

Figure 21-28. 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	Name
R-0h						R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-16. CAN_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 CAN INT1 0 = No interrupt generated 1= Interrupt is generated due to CAN INT1 (refer to CAN Interrupt Status Register for the condition)
0	Name	R	0h	Global Interrupt Flag for CAN INT0 0 = No interrupt generated 1= Interrupt is generated due to CAN INT0 (refer to CAN Interrupt Status Register for the condition)

21.16.2.12 CAN_GLB_INT_CLR Register (Offset = 58h) [reset = 0h]

CAN_GLB_INT_CLR is shown in [Figure 21-29](#) and described in [Table 21-17](#).

CAN Global Interrupt Clear Register

Figure 21-29. 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						R	R
						W-0h	W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-17. 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 CAN INT1 0 = No effect 1= Write '1' to clear the corresponding bit of the Global Interrupt Flag Register
0	INT0_FLG_CLR	W	0h	Global Interrupt flag clear for CAN INT0 0 = No effect 1= Write '1' to clear the corresponding bit of the Global Interrupt Flag Register

21.16.2.13 CAN_ABOTR Register (Offset = 80h) [reset = 0h]

CAN_ABOTR is shown in [Figure 21-30](#) and described in [Table 21-18](#).

Auto-Bus-On Time Register

Figure 21-30. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-18. 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 VBUS clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit.</p> <p>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>

21.16.2.14 CAN_TXRQ_X Register (Offset = 84h) [reset = 0h]

CAN_TXRQ_X is shown in [Figure 21-31](#) and described in [Table 21-19](#).

CAN Transmission Request X Register

Figure 21-31. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-19. 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.
1-0	TxRqstReg1	R	0h	Transmit Request Register 1 flag: With these bits, the CPU can detect if one or more bits in the CAN Transmission Request 2_1 Register (CAN_TXRQ_21) is set. Each register bit represents a group of eight mailboxes. If at least one of the TxRqst bits of these message objects are set, the corresponding bit in the Transmission Request X Register will be set. 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.

21.16.2.15 CAN_TXRQ_21 Register (Offset = 88h) [reset = 0h]

CAN_TXRQ_21 is shown in [Figure 21-32](#) and described in [Table 21-20](#).

CAN Transmission Request 2_1 Register

Figure 21-32. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-20. 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.

21.16.2.16 CAN_NDAT_X Register (Offset = 98h) [reset = 0h]

CAN_NDAT_X is shown in [Figure 21-33](#) and described in [Table 21-21](#).

CAN New Data X Register

Figure 21-33. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-21. 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.
1-0	NewDatReg1	R	0h	New Data Register 1 flag: With these bits, the CPU can detect if one or more bits in the CAN New Data 2_1 Register (CAN_NDAT _21) is set. Each register bit represents a group of eight mailboxes. If at least one of the NewDat bits of these mailboxes are set, the corresponding bit in the CAN New Data X Register will be set. 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.

21.16.2.17 CAN_NDAT_21 Register (Offset = 9Ch) [reset = 0h]

CAN_NDAT_21 is shown in [Figure 21-34](#) and described in [Table 21-22](#).

CAN New Data 2_1 Register

Figure 21-34. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-22. 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>

21.16.2.18 CAN_IPEN_X Register (Offset = ACh) [reset = 0h]

CAN_IPEN_X is shown in [Figure 21-35](#) and described in [Table 21-23](#).

CAN Interrupt Pending X Register

Figure 21-35. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-23. 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.
1-0	IntPndReg1	R	0h	Interrupt Pending Register 1 flag: With these bits, the CPU can detect if one or more bits in the CAN Interrupt Pending 2_1 Register (CAN_IPEN_21) is set. Each register bit represents a group of eight mailboxes. If at least one of the IntPnd bits of these mailboxes are set, the corresponding bit in the CAN Interrupt Pending X Register will be set. 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.

21.16.2.19 CAN_IPEN_21 Register (Offset = B0h) [reset = 0h]

CAN_IPEN_21 is shown in [Figure 21-36](#) and described in [Table 21-24](#).

CAN Interrupt Pending 2_1 Register

Figure 21-36. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-24. 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>

21.16.2.20 CAN_MVAL_X Register (Offset = C0h) [reset = 0h]

CAN_MVAL_X is shown in [Figure 21-37](#) and described in [Table 21-25](#).

CAN Message Valid X Register

Figure 21-37. 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	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-25. 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.
1-0	MsgValReg1	R	0h	Message Valid Register 1 flag: 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 register bit represents a group of eight mailboxes. If at least one of the MsgVal bits of these mailboxes are set, the corresponding bit in the CAN Message Valid X Register will be set. 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.

21.16.2.21 CAN_MVAL_21 Register (Offset = C4h) [reset = 0h]

CAN_MVAL_21 is shown in [Figure 21-38](#) and described in [Table 21-26](#).

CAN Message Valid 2_1 Register

Figure 21-38. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-26. 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>

21.16.2.22 CAN_IP_MUX21 Register (Offset = D8h) [reset = 0h]

CAN_IP_MUX21 is shown in [Figure 21-39](#) and described in [Table 21-27](#).

CAN Interrupt Multiplexer 2_1 Register

Figure 21-39. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-27. CAN_IP_MUX21 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IntMux	R/W	0h	<p>Interrupt Mux bits: The IntMux flag determines for each mailbox, which of the two interrupt lines (CANn_INT0 or CANn_INT1) will be asserted when the IntPnd of this mailbox is set.</p> <p>0 CANn_INT0 line is active if corresponding IntPnd flag is one.</p> <p>1 CANn_INT1 line is active if corresponding IntPnd flag is one.</p>

21.16.2.23 CAN_IF1CMD Register (Offset = 100h) [reset = 1h]

CAN_IF1CMD is shown in [Figure 21-40](#) and described in [Table 21-28](#).

IF1 Command Register

Figure 21-40. 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	RESERVED	RESERVED					
R-0h	R-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-28. 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. 1 Direction = Write: Transfer direction is 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.
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. 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.
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. 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.

Table 21-28. CAN_IF1CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20	Control	R/W	0h	<p>Access Control Bits</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/IF2 Register set.</p> <p>Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/ NewDat bit in the IF1/IF2 Message Control Register will be ignored.</p> <p>Note: This bit is write protected by Busy bit.</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>Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 registers to Message RAM can only be controlled by the Control flag (Bit [20]).</p> <p>Note: This bit is write protected by Busy bit.</p>
18	TXRQST	R/W	0h	<p>Access Transmission Request Bit</p> <p>0 Direction = Read: NewDat bit will not be changed. 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>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>
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>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>
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>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>

Table 21-28. CAN_IF1CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	Busy	R	0h	Busy Flag 0 No transfer between IF1/IF2 Register Set and Message RAM is in progress. 1 Transfer between IF1/IF2 Register Set and Message RAM is in progress. 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.
14	RESERVED	R	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 0x01-0x80 Valid message numbers 0x81-0xFF Invalid message numbers Note: This bit is write protected by Busy bit.

21.16.2.24 CAN_IF1MSK Register (Offset = 104h) [reset = FFFFFFFFh]

CAN_IF1MSK is shown in [Figure 21-41](#) and described in [Table 21-29](#).

IF1 Mask Register

Figure 21-41. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-29. CAN_IF1MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier 0 The extended identifier bit (IDE) has no effect on the acceptance filtering. 1 The extended identifier bit (IDE) 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.
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.
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.

21.16.2.25 CAN_IF1ARB Register (Offset = 108h) [reset = 0h]

CAN_IF1ARB is shown in [Figure 21-42](#) and described in [Table 21-30](#).

IF1 Arbitration Register

Figure 21-42. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-30. CAN_IF1ARB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	<p>Message Valid</p> <p>0 The message object is ignored by the message handler.</p> <p>1 The message object is to be used by the message handler.</p> <p>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.</p> <p>Note: This bit is write protected by Busy bit.</p>
30	Xtd	R/W	0h	<p>Extended Identifier</p> <p>0 The 11-bit ("standard") Identifier is used for this message object.</p> <p>1 The 29-bit ("extended") Identifier is used for this message object.</p> <p>Note: This bit is write protected by Busy bit.</p>
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 message 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.</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.</p>

21.16.2.26 CAN_IF1MCTL Register (Offset = 10Ch) [reset = 0h]

CAN_IF1MCTL is shown in [Figure 21-43](#) and described in [Table 21-31](#).

IF1 Message Control Register

Figure 21-43. 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	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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-31. CAN_IF1MCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R/W	0h	<p>New Data</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: This bit is write protected by Busy bit.</p>
14	MsgLst	R/W	0h	<p>Message Lost (only valid for message objects with direction = receive)</p> <p>0 No message lost since the last time when this bit was reset by the CPU.</p> <p>1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.</p> <p>Note: This bit is write protected by Busy bit.</p>
13	IntPnd	R/W	0h	<p>Interrupt Pending</p> <p>0 This message object is not the source of an interrupt.</p> <p>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.</p> <p>Note: This bit is write protected by Busy bit.</p>
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.</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.</p>

Table 21-31. CAN_IF1MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	RxIE	R/W	0h	Receive Interrupt Enable 0 IntPnd will not be triggered after the successful reception of a frame. 1 IntPnd will be triggered after the successful reception of a frame. Note: This bit is write protected by Busy bit.
9	RmtEn	R/W	0h	Remote Enable 0 At the reception of a remote frame, TxRqst is not changed. 1 At the reception of a remote frame, TxRqst is set. Note: This bit is write protected by Busy bit.
8	TxRqst	R/W	0h	Transmit Request 0 This message object is not waiting for a transmission. 1 The transmission of this message object is requested and is not yet done. Note: This bit is write protected by Busy bit.
7	EoB	R/W	0h	End of Block 0 The message object is part of a FIFO Buffer block and is not the last message object of the 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. Note: This bit is write protected by Busy bit.
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R/W	0h	Data length code 0-8 Data frame has 0-8 data bits. 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.

21.16.2.27 CAN_IF1DATA Register (Offset = 110h) [reset = 0h]

CAN_IF1DATA is shown in [Figure 21-44](#) and described in [Table 21-32](#).

IF1 Data A Register

Figure 21-44. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-32. CAN_IF1DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3
23-16	Data_2	R/W	0h	Data Byte 2
15-8	Data_1	R/W	0h	Data Byte 1
7-0	Data_0	R/W	0h	Data Byte 0 Note: All bits in this register are write protected by Busy bit.

21.16.2.28 CAN_IF1DATB Register (Offset = 114h) [reset = 0h]

CAN_IF1DATB is shown in [Figure 21-45](#) and described in [Table 21-33](#).

IF1 Data B Register

Figure 21-45. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-33. CAN_IF1DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7
23-16	Data_6	R/W	0h	Data Byte 6
15-8	Data_5	R/W	0h	Data Byte 5
7-0	Data_4	R/W	0h	Data Byte 4 Note: All bits in this register are write protected by Busy bit.

21.16.2.29 CAN_IF2CMD Register (Offset = 120h) [reset = 1h]

CAN_IF2CMD is shown in [Figure 21-46](#) and described in [Table 21-34](#).

IF2 Command Register

Figure 21-46. 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	RESERVED	RESERVED					
R-0h	R-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-34. 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. 1 Direction = Write: Transfer direction is 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.
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. 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.
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. 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.

Table 21-34. CAN_IF2CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20	Control	R/W	0h	<p>Access Control Bits</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/IF2 Register set.</p> <p>Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/ NewDat bit in the IF1/IF2 Message Control Register will be ignored.</p> <p>Note: This bit is write protected by Busy bit.</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>Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 registers to Message RAM can only be controlled by the Control flag (Bit [20]).</p> <p>Note: This bit is write protected by Busy bit.</p>
18	TxRqst	R/W	0h	<p>Access Transmission Request Bit</p> <p>0 Direction = Read: NewDat bit will not be changed. 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>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>
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>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>
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>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>

Table 21-34. CAN_IF2CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	Busy	R	0h	Busy Flag 0 No transfer between IF1/IF2 Register Set and Message RAM is in progress. 1 Transfer between IF1/IF2 Register Set and Message RAM is in progress. 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.
14	RESERVED	R	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 0x01-0x80 Valid message numbers 0x81-0xFF Invalid message numbers

21.16.2.30 CAN_IF2MSK Register (Offset = 124h) [reset = FFFFFFFFh]

CAN_IF2MSK is shown in [Figure 21-47](#) and described in [Table 21-35](#).

IF2 Mask Register

Figure 21-47. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-35. CAN_IF2MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier 0 The extended identifier bit (IDE) has no effect on the acceptance filtering. 1 The extended identifier bit (IDE) 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.
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.
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.

21.16.2.31 CAN_IF2ARB Register (Offset = 128h) [reset = 0h]

CAN_IF2ARB is shown in [Figure 21-48](#) and described in [Table 21-36](#).

IF2 Arbitration Register

Figure 21-48. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-36. CAN_IF2ARB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	<p>Message Valid</p> <p>0 The message object is ignored by the message handler.</p> <p>1 The message object is to be used by the message handler.</p> <p>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.</p> <p>Note: This bit is write protected by Busy bit.</p>
30	Xtd	R/W	0h	<p>Extended Identifier</p> <p>0 The 11-bit ("standard") Identifier is used for this message object.</p> <p>1 The 29-bit ("extended") Identifier is used for this message object.</p> <p>Note: This bit is write protected by Busy bit.</p>
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 message 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.</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.</p>

21.16.2.32 CAN_IF2MCTL Register (Offset = 12Ch) [reset = 0h]

CAN_IF2MCTL is shown in [Figure 21-49](#) and described in [Table 21-37](#).

IF2 Message Control Register

Figure 21-49. 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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-37. 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.
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.
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.
12	UMask	R/W	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. Note: This bit is write protected by Busy bit.
11	TxIE	R/W	0h	Transmit Interrupt Enable 0 IntPnd will not be triggered after the successful transmission of a frame. 1 IntPnd will be triggered after the successful transmission of a frame. Note: This bit is write protected by Busy bit.

Table 21-37. CAN_IF2MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	RxIE	R/W	0h	Receive Interrupt Enable 0 IntPnd will not be triggered after the successful reception of a frame. 1 IntPnd will be triggered after the successful reception of a frame. Note: This bit is write protected by Busy bit.
9	RmtEn	R/W	0h	Remote Enable 0 At the reception of a remote frame, TxRqst is not changed. 1 At the reception of a remote frame, TxRqst is set. Note: This bit is write protected by Busy bit.
8	TxRqst	R/W	0h	Transmit Request 0 This message object is not waiting for a transmission. 1 The transmission of this message object is requested and is not yet done. Note: This bit is write protected by Busy bit.
7	EoB	R/W	0h	End of Block 0 The message object is part of a FIFO Buffer block and is not the last message object of the 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. Note: This bit is write protected by Busy bit.
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R/W	0h	Data length code 0-8 Data frame has 0-8 data bits. 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.

21.16.2.33 CAN_IF2DATA Register (Offset = 130h) [reset = 0h]

CAN_IF2DATA is shown in [Figure 21-50](#) and described in [Table 21-38](#).

IF2 Data A Register

Figure 21-50. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-38. CAN_IF2DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3
23-16	Data_2	R/W	0h	Data Byte 2
15-8	Data_1	R/W	0h	Data Byte 1
7-0	Data_0	R/W	0h	Data Byte 0 Note: All bits in this register are write protected by Busy bit.

21.16.2.34 CAN_IF2DATB Register (Offset = 134h) [reset = 0h]

CAN_IF2DATB is shown in [Figure 21-51](#) and described in [Table 21-39](#).

IF2 Data B Register

Figure 21-51. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-39. CAN_IF2DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7
23-16	Data_6	R/W	0h	Data Byte 6
15-8	Data_5	R/W	0h	Data Byte 5
7-0	Data_4	R/W	0h	Data Byte 4 Note: All bits in this register are write protected by Busy bit.

21.16.2.35 CAN_IF3OBS Register (Offset = 140h) [reset = 0h]

CAN_IF3OBS is shown in [Figure 21-52](#) and described in [Table 21-40](#).

IF3 Observation Register

Figure 21-52. 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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-40. 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.
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.
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.
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.
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.
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.
7-5	RESERVED	R	0h	Reserved

Table 21-40. CAN_IF3OBS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	Data_B	R/W	0h	Data B read observation 0 Data B section has not to be read. 1 Data B section has to be read to enable next IF3 update.
3	Data_A	R/W	0h	Data A read observation 0 Data A section has not to be read. 1 Data A section has to be read to enable next IF3 update.
2	Ctrl	R/W	0h	Ctrl read observation 0 Ctrl section has not to be read. 1 Ctrl section has to be read to enable next IF3 update.
1	Arb	R/W	0h	Arbitration data read observation 0 Arbitration data has not to be read. 1 Arbitration data has to be read to enable next IF3 update.
0	Mask	R/W	0h	Mask data read observation 0 Mask data has not to be read. 1 Mask data has to be read to enable next IF3 update.

21.16.2.36 CAN_IF3MSK Register (Offset = 144h) [reset = FFFFFFFFh]

CAN_IF3MSK is shown in [Figure 21-53](#) and described in [Table 21-41](#).

IF3 Mask Register

Figure 21-53. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-41. CAN_IF3MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd	R	1h	Mask Extended Identifier 0 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.
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.
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

21.16.2.37 CAN_IF3ARB Register (Offset = 148h) [reset = 0h]

CAN_IF3ARB is shown in [Figure 21-54](#) and described in [Table 21-42](#).

IF3 Arbitration Register

Figure 21-54. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-42. 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.
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.
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).
28-0	ID	R	0h	Message Identifier ID[28:0] 29-bit Identifier ("Extended Frame") ID[28:18] 11-bit Identifier ("Standard Frame")

21.16.2.38 CAN_IF3MCTL Register (Offset = 14Ch) [reset = 0h]

CAN_IF3MCTL is shown in [Figure 21-55](#) and described in [Table 21-43](#).

IF3 Message Control Register

Figure 21-55. 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	TxIE	RxIE	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			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-43. 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.
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.
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.
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.
11	TxIE	R	0h	Transmit Interrupt Enable 0 IntPnd will not be triggered after the successful transmission of a frame. 1 IntPnd will be triggered after the successful transmission of a frame.
10	RxIE	R	0h	Receive Interrupt Enable 0 IntPnd will not be triggered after the successful reception of a frame. 1 IntPnd will be triggered after the successful reception of a frame.

Table 21-43. CAN_IF3MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	RmtEn	R	0h	Remote Enable 0 At the reception of a remote frame, TxRqst is not changed. 1 At the reception of a remote frame, TxRqst is set.
8	TxRqst	R	0h	Transmit Request 0 This message object is not waiting for a transmission. 1 The transmission of this message object is requested and is not yet done.
7	EoB	R	0h	End of Block 0 The message object is part of a FIFO Buffer block and is not the last message object of the 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.
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R	0h	Data length code 0-8 Data frame has 0-8 data bits. 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.

21.16.2.39 CAN_IF3DATA Register (Offset = 150h) [reset = 0h]

CAN_IF3DATA is shown in [Figure 21-56](#) and described in [Table 21-44](#).

IF3 Data A Register

Figure 21-56. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-44. CAN_IF3DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3	R	0h	Data Byte 3
23-16	Data_2	R	0h	Data Byte 2
15-8	Data_1	R	0h	Data Byte 1
7-0	Data_0	R	0h	Data Byte 0

21.16.2.40 CAN_IF3DATB Register (Offset = 154h) [reset = 0h]

CAN_IF3DATB is shown in [Figure 21-57](#) and described in [Table 21-45](#).

IF3 Data B Register

Figure 21-57. 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							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-45. CAN_IF3DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7	R	0h	Data Byte 7
23-16	Data_6	R	0h	Data Byte 6
15-8	Data_5	R	0h	Data Byte 5
7-0	Data_4	R	0h	Data Byte 4

21.16.2.41 CAN_IF3UPD Register (Offset = 160h) [reset = 0h]

CAN_IF3UPD is shown in [Figure 21-58](#) and described in [Table 21-46](#).

IF3 Update Enable Register

Figure 21-58. 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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 21-46. CAN_IF3UPD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IF3UpdEn	R/W	0h	IF3 Update Enabled (for all message objects) 0 Automatic IF3 update is disabled for this message object. 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.

Universal Serial Bus (USB) Controller

This chapter discusses the features and functions of the universal serial bus (USB) controller.

Topic	Page
22.1 Introduction	2187
22.2 Features	2187
22.3 Functional Description	2189
22.4 Initialization and Configuration	2198
22.5 Register Map	2199
22.6 Register Descriptions	2204

22.1 Introduction

The USB controller operates as a full-speed function controller during point-to-point communications with the USB host. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. It has thirty-two endpoints, sixteen for IN transactions and sixteen for OUT transactions. One IN and one OUT endpoint are fixed-function endpoints used for control transfers; the others are defined by firmware. A dynamically sizeable FIFO supports queuing multiple packets. Software-controlled connect and disconnect allow flexibility during USB device startup.

22.2 Features

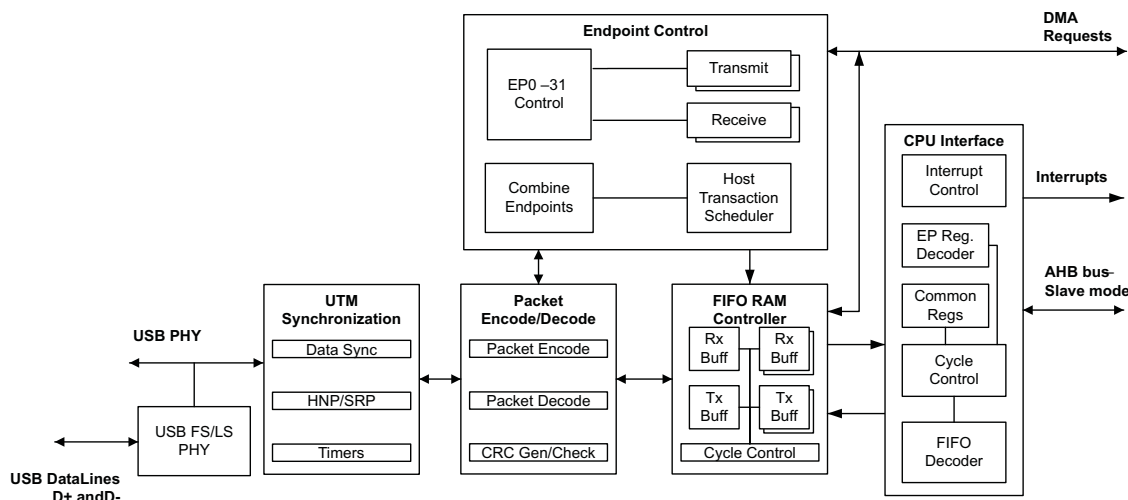
The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) operation in host and device modes as well as low-speed (1.5 Mbps) operation in host mode
- Integrated PHY
- Four transfer types: Control, Interrupt, Bulk, and Isochronous
- Thirty-two endpoints
 - One dedicated control IN endpoint and one dedicated control OUT endpoint
 - Fifteen configurable IN endpoints and fifteen configurable OUT endpoints
- Four KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size

22.2.1 Block Diagram

The USB block diagram is shown in [Figure 22-1](#).

Figure 22-1. USB Block Diagram



22.2.2 Signal Description

The USB controller requires a total of three signals (D+, D-, and V_{BUS}) to operate in device mode and two signals (D+, D-) to operate in embedded host mode. Because of the differential signaling needed for USB, the pins D+ and D- have special buffers to support USB. As such, their position on the chip is not user-selectable. These pins at reset are, by default, GPIOs. They must be configured before being used as USB function pins. Bits 10 and 11 in the GPIO B Analog Mode Select register (GPBAMSEL) should be set to choose the USB function. The signals USB bus voltage (V_{BUS}), external power enable (EPEN), and power fault (PFLT) are not hardwired to any pin and some applications will require they be implemented in software via a GPIO. Software that implements these signals is available in the USB software library.

22.2.3 VBus Recommendations

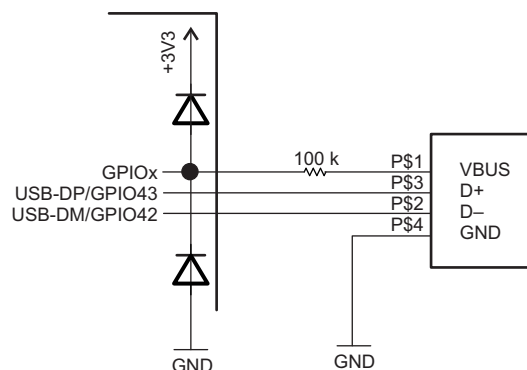
Section 22.2.3 Most applications do not need to monitor V_{BUS} . Because of this, a dedicated V_{BUS} monitoring pin was not included on this microcontroller. If you are designing a bus-powered device application or an embedded host application, you do NOT need to monitor V_{BUS} . If you are designing a self-powered device, you will need to actively monitor the state of the V_{BUS} pin in order to ensure compliance with the USB specification. In Section 7.1.5 and Section 7.2.1 of the USB Specification Revision 2.0™ it is stated respectively that:

- "The voltage source on the [speed identification] pull-up resistor must be derived from or controlled by the power supplied on the USB cable such that when V_{BUS} is removed, the pull-up resistor does not supply current on the data line to which it is attached.
- When V_{BUS} is removed, the device must remove power from the D+/D- pull-up resistor within 10 seconds.
- Later in the timing tables (Section 7.3.2) of the USB Specification 2.0 it is also stated that the D+/D- pull-up resistor should be applied within 100ms of V_{BUS} reaching a valid level."

Meeting the above specification is easy because of the slow timing requirements. In this chapter we will discuss the hardware part of the V_{BUS} monitoring solution. The corresponding software will be discussed briefly, but for examples and an explanation, please consult the USB software guide.

The pins of this microcontroller are NOT 5V tolerant, and because of this the V_{BUS} signal cannot be directly connected to a GPIO pin. Directly connecting 5V to a pin of the microcontroller WILL destroy the I/O buffer of the pin and possibly more of the chip. The most cost-effective way of making any pin capable of reading a 5V input is to use a series resistance in conjunction with the ESD diode clamps already present inside the device on every pin. We recommend the use of a 100kΩ series resistor between the V_{BUS} signal and the pin chosen to monitor it. A diagram of this setup is shown below.

Figure 22-2. USB Scheme



In the above diagram, if V_{BUS} is above or below 3.3V and 0V respectively, one of the ESD clamp diodes will be forward-biased, allowing current to flow through the 100kΩ resistor. The purpose of the diode clamps is to protect the pins of the microcontroller from very short overvoltage spikes of a high magnitude. They do this by clamping the voltage excursion to one of the supply rails. We are effectively requiring the ESD clamps to do the same thing they were designed to do, but instead of a short high magnitude pulse, we are giving them a long low magnitude static value via the 100kΩ resistor.

Any pin that has digital input/output functionality could potentially be used to monitor V_{BUS} , but the use of an interrupt capable GPIO is recommended. A pin that does not have external interrupt capability may also be used, but the input state of the pin must be polled periodically by the application software to ensure appropriate action is taken whenever V_{BUS} is applied or removed. If an interrupt capable GPIO is chosen, it should be configured to generate an interrupt on both the rising and falling edge. More information on external interrupts can be found in the *System Control and Interrupts* chapter. Example code that implements V_{BUS} monitoring using external interrupts and takes the appropriate actions is documented in the USB Software Guide and can be found in the associated USB software package.

22.3 Functional Description

The USB controller can be configured to act as either a dedicated host or device. However, when the USB controller is acting as a self-powered device, a GPIO input or analog comparator input must be connected to V_{BUS} and configured to generate an interrupt when the V_{BUS} level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

Note: [Section 22.3](#) When USB is used in the system, the minimum system frequency is 30 MHz.

22.3.1 Operation as a Device

This section describes how the USB controller performs when it is being used as a USB device. IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of start of frame (SOF) are all described.

When in device mode, IN transactions are controlled by the endpoint transmit interface and uses the transmit endpoint registers for the given endpoint. OUT transactions are handled with the endpoints receive interface and uses the receive endpoint registers for the given endpoint. When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- Bulk. Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt. Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- Isochronous. Isochronous endpoints are more flexible and can be up to 1023 bytes.
- Control. It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device should use the dedicated control endpoint on the USB controller's endpoint 0.

22.3.1.1 Control and Configurable Endpoints

When operating as a device, the USB controller provides two dedicated control endpoints (IN and OUT) and six configurable endpoints (three IN and three OUT) that can be used for communications with a host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface. Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions. The remaining six endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as three configurable IN and three configurable OUT endpoints. The endpoint pairs are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

22.3.1.1.1 IN Transactions as a Device

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the three configurable IN endpoints are determined by the USB Transmit FIFO Start Address (USBTXFIFOADD) register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the USB Maximum Transmit Data Endpoint n (USBTXMAXPn) register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The USBTXMAXPn register should not be written to while data is in the FIFO as unexpected results may occur.

Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ) register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the TXRDY bit in the USB Transmit Control and Status Endpoint n Low (USBTXCSSLn) register must be set. If the AUTOSET bit in the USB Transmit Control and Status Endpoint n High (USBTXCSSLn) register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the TXRDY bit must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both TXRDY and FIFONE are cleared, and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the TXRDY bit in the USBTXCSSLn register must be set. If the AUTOSET bit in the USBTXCSSLn register is set, the TXRDY bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TXRDY must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, TXRDY is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and TXRDY set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, TXRDY is automatically cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the FIFONE bit in the USBTXCSSLn register at this point indicates how many packets may be loaded. If the FIFONE bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then no packets are in the FIFO and two more packets can be loaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS) register. This bit is set by default, so it must be cleared to enable double-packet buffering.

22.3.1.1.2 Out Transactions as a Device

When in device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the three configurable OUT endpoints are determined by the USB Receive FIFO Start Address (USBRXFIFOADD) register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the USB Maximum Receive Data Endpoint n (USBRXMAXPn) register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

Note: In all cases, the maximum packet size must not exceed the FIFO size.

Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the USB Receive Control and Status Endpoint n Low (USBXCSRL[n]) register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RXRDY bit must be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the Host controller. If the AUTOCL bit in the USB Receive Control and Status Endpoint n High (USBXCSRH[n]) register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY and FULL bits are cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually.

Double-Packet Buffering

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the RXRDY bit in the USBXCSRL[n] register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

Note: The FULL bit in USBXCSRL[n] is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the RXRDY bit must be cleared to allow further packets to be received. If the AUTOCL bit in the USBXCSRH[n] register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY bit is cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually. If the FULL bit is set when RXRDY is cleared, the USB controller first clears the FULL bit, then sets RXRDY again to indicate that there is another packet waiting in the FIFO to be unloaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the USB Receive Double Packet Buffer Disable (USBXDPKTBUFDIS) register. This bit is set by default, so it must be cleared to enable double-packet buffering.

22.3.1.1.3 Scheduling

The device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The USB controller can set up a transaction at any time. The USB controller waits for the request from the Host controller and generates an interrupt when the transaction is complete or if it was terminated due to some error. If the Host controller makes a request and the device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

22.3.1.1.4 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller such as when the USB controller automatically stalls a control transfer or unexpected zero length OUT data packets.

Stalled Control Transfer

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The Host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the DATAEND bit in the USB Control and Status Endpoint 0 Low (USBCSRL0) register has been set.
2. The Host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an IN token (instead of an OUT token) after the CPU has cleared TXRDY and set DATAEND in response to the ACK issued by the Host to what should have been the last packet.
3. The Host sends more than USBRXMAXPn bytes of data with an OUT data token.
4. The Host sends more than a zero length data packet for the OUT STATUS phase.

Zero Length OUT Data Packets

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred. However, if the Host sends a zero-length OUT data packet before the entire length of device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the DATAEND bit in the USBCSRLO register.

Setting the Device Address

When a Host is attempting to enumerate the USB device, it requests that the device change its address from zero to some other value. The address is changed by writing the value that the Host requested to the USB Device Functional Address (USBFADDR) register. However, care should be taken when writing to USBFADDR to avoid changing the address before the transaction is complete. This register should only be set after the SET_ADDRESS command is complete. Like all control transactions, the transaction is only complete after the device has left the STATUS phase. In the case of a SET_ADDRESS command, the transaction is completed by responding to the IN request from the Host with a zero-byte packet. Once the device has responded to the IN request, the USBFADDR register should be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

Note: If the USBFADDR register is set to the new value as soon as the device receives the OUT transaction with the SET_ADDRESS command in the packet, it changes the address during the control transfer. In this case, the device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the Host does not get a response to the IN request, and the Host fails to enumerate the device.

22.3.1.1.5 Device Mode Suspend

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the USB Interrupt Enable (USBIE) register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the RESUME bit in the USB Power (USBPOWER) register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling. To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state.

22.3.1.1.6 Start of Frame

When the USB controller is operating in device mode, it receives a Start-Of-Frame (SOF) packet from the Host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the USB Frame Value (USBFRAME) register, and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, the packet is assumed to have been lost, and the USBFRAME register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

22.3.1.1.7 USB Reset

When the USB controller is in device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the USBFADDR register
- Clears the USB Endpoint Index (USBEPIDX) register
- Flushes all endpoint FIFOs
- Clears all control/status registers
- Enables all endpoint interrupts

- Generates a RESET interrupt

22.3.1.1.8 Connect/Disconnect

The USB controller connection to the USB bus is handled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the SOFTCONN bit of the USBPOWER register. When the SOFTCONN bit is set, the PHY is placed in its normal mode, and the USB0DP/USB0DM lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it does not respond to any USB signaling except a USB RESET. When the SOFTCONN bit is cleared, the PHY is put into non-driving mode, USB0DP and USB0DM are tristated, and the USB controller appears to other devices on the USB bus as if it has been disconnected. The non-driving mode is the default so the USB controller appears disconnected until the SOFTCONN bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete, and the system is ready to perform enumeration before connecting to the USB bus. Once the SOFTCONN bit has been set, the USB controller can be disconnected by clearing this bit.

Note: The USB controller does not generate an interrupt when the device is connected to the Host. However, an interrupt is generated when the Host terminates a session.

22.3.2 Operation as a Host

When the USB controller is operating in Host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB Host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in Host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- Bulk. Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- Interrupt. Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- Isochronous. Isochronous endpoints are more flexible and can be up to 1023 bytes.
- Control. It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a device's endpoint 0.

22.3.2.1 Endpoint Registers

The endpoint registers are used to control the USB endpoint interfaces which communicate with device(s) that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, 3 configurable OUT endpoints, and 3 configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous device endpoints.

These USB interfaces can be used to simultaneously schedule as many as 3 independent OUT and 3 independent IN transactions to any endpoints on any device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a device's bulk OUT endpoint 1, while the IN portion is communicating with a device's interrupt IN endpoint 2.

Before accessing any device, whether for point-to-point communications or for communications via a hub, the relevant USB Receive Functional Address Endpoint *n* (USBRXFUNCADDR_{*n*}) or USB Transmit Functional Address Endpoint *n* (USBTXFUNCADDR_{*n*}) registers must be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

22.3.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in device mode except that the transaction first must be initiated by setting the REQPKT bit in the USBCSRL0 register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target device. When the packet is received and placed in the receive FIFO, the RXRDY bit in the USBCSRL0 register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY must be cleared. The AUTOCL bit in the USBRXCSR_{*Hn*} register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The AUTORQ bit in USBRXCSR_{*Hn*} causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. When the RXRDY bit is cleared, the controller sends an acknowledge to the device. When there is a known number of packets to be transferred, the USB Request Packet Count in Block Transfer Endpoint *n* (USBRQPKTCOUNT_{*n*}) register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the USBRQPKTCOUNT_{*n*} register following each request. When the USBRQPKTCOUNT_{*n*} value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, USBRQPKTCOUNT_{*n*} should be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the USBRXMAXP_{*n*} register) such as may occur at the end of a bulk transfer.

If the device responds to a bulk or interrupt IN token with a NAK, the USB Host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target device responds with a STALL, however, the USB Host controller does not retry the transaction but sets the STALLED bit in the USBCSRL0 register. If the target device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target device has still not responded, the USB Host controller clears the REQPKT bit and sets the ERROR bit in the USBCSRL0 register.

22.3.2.3 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in device mode. The TXRDY bit in the USBTXCSRL_{*n*} register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the USBTXCSR_{*Hn*} register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO.

If the target device responds to the OUT token with a NAK, the USB Host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the USBTXCSRL_{*n*} register. If the target device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and sets the ERROR bit in the USBTXCSRL_{*n*} register.

22.3.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB Host controller. The Host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding. Isochronous endpoints can be scheduled from every frame to every 216 frames, in powers of 2.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, a K state is transmitted on the bus to act as a keep-alive to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB Host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and/or the FIFONE bit is set.

An isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the USB Host Transmit Interval Endpoint n (USBTXINTERVAL[n]) or USB Host Receive Interval Endpoint n (USBRXINTERVAL[n]) register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.

22.3.2.5 USB Hubs

The following setup requirements apply to the USB Host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding USB Receive Hub Address Endpoint n (USBRXHUBADDR n) and USB Receive Hub Port Endpoint n (USBRXHUBPORT n) or the USB Transmit Hub Address Endpoint n (USBTXHUBADDR n) and USB Transmit Hub Port Endpoint n (USBTXHUBPORT n) registers. In addition, the speed at which the device operates (full or low) must be recorded in the USB Type Endpoint 0 (USBTYP0) (endpoint 0), USB Host Configure Transmit Type Endpoint n (USBTXTYP n), or USB Host Configure Receive Type Endpoint n (USBRXTYP n) registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB Host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions must be made following the completion of any on-going transactions on the endpoints affected.

22.3.2.6 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB Host controller assumes that the target device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

22.3.2.7 Host SUSPEND

If the SUSPEND bit in the USBPOWER register is set, the USB Host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the RESUME bit and clear the SUSPEND bit. While the RESUME bit is set, the USB Host controller generates RESUME signaling on the bus. After 20 ms, the RESUME bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

22.3.2.8 USB RESET

If the RESET bit in the USBPOWER register is set, the USB Host controller generates USB RESET signaling on the bus. The RESET bit must be set for at least 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the USB Host controller starts its frame counter and transaction scheduler.

22.3.2.9 Connect/Disconnect

A session is started by setting the SESSION bit in the USB device Control (USBDEVCTL) register, enabling the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the USBDEVCTL register where the FSDEV bit is set for a full-speed device, and the LSDEV bit is set for a low-speed device. The USB controller must generate a RESET to the device, and then the USB Host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.

22.3.3 DMA Operation

The USB module's DMA trigger signals are not supported on this device. The DMA controller may be used to read and write the USB FIFOs via software triggering. see the Direct Memory Access (DMA) chapter for more details about programming the DMA controller.

22.3.4 Address/Data Bus Bridge

The USB controller on this device is the same controller that is on the Stellaris devices. This controller was originally designed to connect to an ARM AHB bus, but has been modified in order to function with the C28x device's bus architecture. The modifications made are largely invisible to the user application, but there are some things to note.

- The USB memory space is 8 bits wide, while the C28x memory space is 16 bits wide.
- 32 and 16 bit accesses (r/w) are completely transparent to the user application code, no changes need be made.
- The C28x core only supports 8 bit accesses through a byte intrinsic type. This can be used to perform 8 bit reads or writes to the USB controller.
 - `int &__byte(int *array, unsigned int byte_index);`
 - `*array = ptr to address to access, byte_index = always 0 (for USB)`
See [Table 22-1](#) for example.
 - See the [TMS320C28x Optimizing C/C++ Compiler User's Guide \(SPRU514\)](#) and the [TMS320C28x Assembly Language Tools User's Guide \(SPRU513\)](#)
- Because of the bridge, the memory view of the USB controller memory space in CCS isn't a 1:1 representation of what is in the controller
 - When the view mode is
 - 32 bit or 16 bit, even address are effectively duplicated, ignore odd addresses.
 - 8 bit, Even addresses from within the controller are duplicated into odd address in the view window; odd addresses from within the controller are not displayed.
See [Table 22-2](#) for example.

Table 22-1. USB Memory Access From Software

USB Controller Memory			C28x 8 Bit	
Address	Reg. Name	Data	Access	Data
0x00	FADDR	0x00	<code>__byte((int *)0x00,0)</code>	0x0000

Table 22-1. USB Memory Access From Software (continued)

USB Controller Memory			C28x 8 Bit	
0x01	POWER	0x11	__byte((int *)0x01,0)	0x0011
0x02	TXIS (LSB)	0x22	__byte((int *)0x02,0)	0x0022
0x03	TXIS (MSB)	0x33	__byte((int *)0x03,0)	0x0033
0x04	RXIS (LSB)	0x44	__byte((int *)0x04,0)	0x0044
0x05	RXIS (MSB)	0x55	__byte((int *)0x05,0)	0x0055
0x06	TXIE (LSB)	0x66	__byte((int *)0x06,0)	0x0066
0x07	TXIE (MSB)	0x77	__byte((int *)0x07,0)	0x0077
0x08	RXIE (LSB)	0x88	__byte((int *)0x08,0)	0x0088
0x09	RXIE (MSB)	0x99	__byte((int *)0x09,0)	0x0099
0x0A	USBIS	0xAA	__byte((int *)0x0A,0)	0x00AA
0x0B	USBIE	0xBB	__byte((int *)0x0B,0)	0x00BB
0x0C	FRAME (LSB)	0xCC	__byte((int *)0x0C,0)	0x00CC
0x0D	FRAME (MSB)	0xDD	__byte((int *)0x0D,0)	0x00DD
0x0E	EPIDX	0xEE	__byte((int *)0x0E,0)	0x00EE
0x0F	TEST	0xFF	__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
((short *)0x0F))	0xFFEE		((long *)0x0F))	0xFFEEDDCC

Table 22-2. USB Memory 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	0xBBAABBA
0x06	0x66	0x06	0x7766	0x0C	0xDDCCDDCC
0x07	0x66	0x07	0x7766	0x0E	0xFFEEFFEE
0x08	0x88	0x08	0x9988		

Table 22-2. USB Memory Access From CCS (continued)

CCS 8 Bit		CCS 16 Bit		CCS 32 Bit	
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		

22.4 Initialization and Configuration

To use the USB controller, the peripheral clock must be enabled via the System Control module's PCLKCR11 register. In addition, the USB PHY signals must be connected to their respective pins via the GPIO module's GPBAMSEL register. Set bits 10 and 11 for USB0DM (GPIO42) and USB0DP (GPIO43).

Set up the auxiliary PLL so a 60 MHz output clock is provided to the USB module. This fixed frequency is required for all USB operations. See the *System Control* chapter for more details.

In host mode, the USB controller is responsible for supplying power to the bus. To avoid incorrectly supplying voltage to the bus, the external power control signal, USB0EPEN, should be kept inactive on start-up. This can be done by connecting the USB0EPEN and USB0PFLT pins to the USB controller as soon as possible.

22.4.1 Pin Configuration

In order to give the user more flexibility, the signals External Power Enable (EPEN) and Power Fault (PFLT) were not implemented in hardware. Instead, it is left up to the user to implement these signals in software. Examples of how to implement these signals in software can be found in the F2806x USB Software Guide.

When using the device controller portion of the USB controller in a system that also provides host functionality, the power to V_{BUS} must be disabled to allow the external host controller to supply power. Usually, the EPEN signal is used to control the external regulator and should be negated to avoid having two devices driving the V_{BUS} power pin on the USB connector.

When the USB controller is acting as a host, it is in control of two signals that are attached to an external voltage supply that provides power to V_{BUS} . The Host controller uses the EPEN signal to enable or disable power to the V_{BUS} pin on the USB connector. An input pin, PFLT, provides feedback when there has been a power fault on V_{BUS} . The PFLT signal can be configured to either automatically negate the EPEN signal to disable power, and/or it can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both EPEN and PFLT are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the Host controller code to respond to these external events.

22.4.2 Endpoint Configuration

To start communication in Host or device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a device. In device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. In device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the Host controller. In Host mode, the endpoints must

be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the USB device controller's soft connect must be enabled when the device is ready to start communications, indicating to the host controller that the device is ready to start the enumeration process. If operating as a Host controller, the device soft connect must be disabled and power must be provided to V_{BUS} via the USB0EPEN signal.

22.5 Register Map

[Table 22-3](#) lists the registers. All addresses given are relative to the USB base address of 0x40000. Note that the USB controller clock must be enabled before the registers can be programmed (see the *System Control* chapter).

Table 22-3. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	Location
0x000	USBFADDR ⁽¹⁾	R/W	0x00	USB Device Functional Address	Section 22.6.1
0x001	USBPOWER ⁽¹⁾⁽²⁾	R/W	0x20	USB Power	Section 22.6.2
0x002	USBTXIS ⁽¹⁾⁽²⁾	RO	0x0000	USB Transmit Interrupt Status	Section 22.6.3
0x004	USBRXIS ⁽¹⁾⁽²⁾	RO	0x0000	USB Receive Interrupt Status	Section 22.6.4
0x006	USBTXIE ⁽¹⁾⁽²⁾	R/W	0xFFFF	USB Transmit Interrupt Enable	Section 22.6.5
0x008	USBRXIE ⁽¹⁾⁽²⁾	R/W	0xFFFE	USB Receive Interrupt Enable	Section 22.6.6
0x00A	USBIS ⁽¹⁾⁽²⁾	RO	0x00	USB General Interrupt Status	Section 22.6.7
0x00B	USBIE ⁽¹⁾⁽²⁾	R/W	0x06	USB Interrupt Enable	Section 22.6.8
0x00C	USBFRAME ⁽¹⁾⁽²⁾	RO	0x0000	USB Frame Value	Section 22.6.9
0x00E	USBEPID ⁽¹⁾⁽²⁾	R/W	0x00	USB Endpoint Index	Section 22.6.10
0x00F	USBTEST ⁽¹⁾⁽²⁾	R/W	0x00	USB Test Mode	Section 22.6.11
0x020	USBFIFO0 ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB FIFO Endpoint 0	Section 22.6.12
0x024	USBFIFO1 ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB FIFO Endpoint 1	Section 22.6.12
0x028	USBFIFO2 ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB FIFO Endpoint 2	Section 22.6.12
0x02C	USBFIFO3 ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB FIFO Endpoint 3	Section 22.6.12
0x060	USBDEVCTL ⁽²⁾	R/W	0x80	USB Device Control	Section 22.6.13
0x062	USBTXFIFOSZ ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Dynamic FIFO Sizing	Section 22.6.14
0x063	USBRXFIFOSZ ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Dynamic FIFO Sizing	Section 22.6.15
0x064	USBTXFIFOADD ⁽¹⁾⁽²⁾	R/W	0x0000	USB Transmit FIFO Start Address	Section 22.6.16
0x066	USBRXFIFOADD ⁽¹⁾⁽²⁾	R/W	0x0000	USB Receive FIFO Start Address	Section 22.6.17
0x07A	USBCONTIM ⁽¹⁾⁽²⁾	R/W	0x5C	USB Connect Timing	Section 22.6.18

Table 22-3. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Location
0x07D	USBFSEOF ⁽¹⁾⁽²⁾	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	Section 22.6.19
0x07E	USBLSEOF ⁽¹⁾⁽²⁾	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	Section 22.6.20
0x080	USBTXFUNCADDR0 ⁽²⁾	R/W	0x00	USB Transmit Functional Address Endpoint 0	Section 22.6.21
0x082	USBTXHUBADDR0 ⁽²⁾	R/W	0x00	USB Transmit Hub Address Endpoint 0	Section 22.6.22
0x083	USBTXHUBPORT0 ⁽²⁾	R/W	0x00	USB Transmit Hub Port Endpoint 0	Section 22.6.23
0x088	USBTXFUNCADDR1 ⁽²⁾	R/W	0x00	USB Transmit Functional Address Endpoint 1	Section 22.6.21
0x08A	USBTXHUBADDR1 ⁽²⁾	R/W	0x00	USB Transmit Hub Address Endpoint 1	Section 22.6.22
0x08B	USBTXHUBPORT1 ⁽²⁾	R/W	0x00	USB Transmit Hub Port Endpoint 1	Section 22.6.23
0x08C	USBRXFUNCADDR1 ⁽²⁾	R/W	0x00	USB Receive Functional Address Endpoint 1	Section 22.6.24
0x08E	USBRXHUBADDR1 ⁽²⁾	R/W	0x00	USB Receive Hub Address Endpoint 1	Section 22.6.25
0x08F	USBRXHUBPORT1 ⁽²⁾	R/W	0x00	USB Receive Hub Port Endpoint 1	Section 22.6.26
0x090	USBTXFUNCADDR2 ⁽²⁾	R/W	0x00	USB Transmit Functional Address Endpoint 2	Section 22.6.21
0x092	USBTXHUBADDR2 ⁽²⁾	R/W	0x00	USB Transmit Hub Address Endpoint 2	Section 22.6.22
0x093	USBTXHUBPORT2 ⁽²⁾	R/W	0x00	USB Transmit Hub Port Endpoint 2	Section 22.6.23
0x094	USBRXFUNCADDR2 ⁽²⁾	R/W	0x00	USB Receive Functional Address Endpoint 2	Section 22.6.24
0x096	USBRXHUBADDR2 ⁽²⁾	R/W	0x00	USB Receive Hub Address Endpoint 2	Section 22.6.25
0x097	USBRXHUBPORT2 ⁽²⁾	R/W	0x00	USB Receive Hub Port Endpoint 2	Section 22.6.26
0x098	USBTXFUNCADDR3 ⁽²⁾	R/W	0x00	USB Transmit Functional Address Endpoint 3	Section 22.6.21
0x09A	USBTXHUBADDR3 ⁽²⁾	R/W	0x00	USB Transmit Hub Address Endpoint 3	Section 22.6.22
0x09B	USBTXHUBPORT3 ⁽²⁾	R/W	0x00	USB Transmit Hub Port Endpoint 3	Section 22.6.23
0x09C	USBRXFUNCADDR3 ⁽²⁾	R/W	0x00	USB Receive Functional Address Endpoint 3	Section 22.6.24
0x09E	USBRXHUBADDR3 ⁽²⁾	R/W	0x00	USB Receive Hub Address Endpoint 3	Section 22.6.25
0x09F	USBRXHUBPORT3 ⁽²⁾	R/W	0x00	USB Receive Hub Port Endpoint 3	Section 22.6.26
0x102	USBCSRLO ⁽¹⁾⁽²⁾	W1C	0x00	USB Control and Status Endpoint 0 Low	Section 22.6.28

Table 22-3. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Location
0x103	USBCSRH0 ⁽¹⁾⁽²⁾	W1C	0x00	USB Control and Status Endpoint 0 High	Section 22.6.29
0x108	USBCOUNT0 ⁽¹⁾⁽²⁾	R/o	0x00	USB Receive Byte Count Endpoint 0	Section 22.6.30
0x10A	USBTYPE0 ⁽²⁾	R/W	0x00	USB Type Endpoint 0	Section 22.6.31
0x10B	USBNAKLMT ⁽²⁾	R/W	0x00	USB NAK Limit	Section 22.6.32
0x110	USBTXMAXP1 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	Section 22.6.27
0x112	USBTXCURL1 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	Section 22.6.33
0x113	USBTXCSRH1 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	Section 22.6.34
0x114	USBRXMAXP1 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Receive Data Endpoint 1	Section 22.6.35
0x116	USBRXCURL1 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	Section 22.6.36
0x117	USBRXCSRH1 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 1 High	Section 22.6.37
0x118	USBRXCOUNT1 ⁽¹⁾⁽²⁾	RO	0x0000	USB Receive Byte Count Endpoint 1	Section 22.6.38
0x11A	USBTXTYPE1 ⁽²⁾	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	Section 22.6.39
0x11B	USBTXINTERVAL1 ⁽²⁾	R/W	0x00	USB Host Transmit Interval Endpoint 1	Section 22.6.40
0x11C	USBRXTYPE1 ⁽²⁾	R/W	0x00	USB Host Configure Receive Type Endpoint 1	Section 22.6.41
0x11D	USBRXINTERVAL1 ⁽²⁾	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	Section 22.6.42
0x120	USBTXMAXP2 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	Section 22.6.27
0x122	USBTXCURL2 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	Section 22.6.33
0x123	USBTXCSRH2 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	Section 22.6.34
0x124	USBRXMAXP2 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Receive Data Endpoint 2	Section 22.6.35
0x126	USBRXCURL2 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	Section 22.6.36
0x127	USBRXCSRH2 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 2 High	Section 22.6.37
0x128	USBRXCOUNT2 ⁽¹⁾⁽²⁾	RO	0x0000	USB Receive Byte Count Endpoint 2	Section 22.6.38

Table 22-3. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Location
0x12A	USBTXTYPE2 ⁽²⁾	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	Section 22.6.39
0x12B	USBTXINTERVAL2 ⁽²⁾	R/W	0x00	USB Host Transmit Interval Endpoint 2	Section 22.6.40
0x12C	USBRXTYPE2 ⁽²⁾	R/W	0x00	USB Host Configure Receive Type Endpoint 2	Section 22.6.41
0x12D	USBRXINTERVAL2 ⁽²⁾	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	Section 22.6.42
0x130	USBTXMAXP3 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	Section 22.6.27
0x132	USBTXCSSL3 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	Section 22.6.33
0x133	USBTXCSRH3 ⁽¹⁾⁽²⁾	R/W	0x00	USB Transmit Control and Status Endpoint 3 High Section 22.6.34	
0x134	USBRXMAXP3 ⁽¹⁾⁽²⁾	R/W	0x0000	USB Maximum Receive Data Endpoint 3	Section 22.6.35
0x136	USBRXCSSL3 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	Section 22.6.33
0x137	USBRXCSRH3 ⁽¹⁾⁽²⁾	R/W	0x00	USB Receive Control and Status Endpoint 3 High	Section 22.6.36
0x138	USBRXCOUNT3 ⁽¹⁾⁽²⁾	RO	0x0000	USB Receive Byte Count Endpoint 3	Section 22.6.38
0x13A	USBTXTYPE3 ⁽²⁾	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	Section 22.6.39
0x13B	USBTXINTERVAL3 ⁽²⁾	R/W	0x00	USB Host Transmit Interval Endpoint 3	Section 22.6.40
0x13C	USBRXTYPE3 ⁽²⁾	R/W	0x00	USB Host Configure Receive Type Endpoint 3	Section 22.6.41
0x13D	USBRXINTERVAL3 ⁽²⁾	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	Section 22.6.42
0x304	USBRQPKTCOUNT1 ⁽²⁾	R/W	0x0000 1	USB Request Packet Count in Block Transfer Endpoint 1	Section 22.6.43
0x308	USBRQPKTCOUNT2 ⁽²⁾	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	Section 22.6.43
0x30C	USBRQPKTCOUNT3 ⁽²⁾	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	Section 22.6.43
0x340	USBRXDPKTBUFFDIS ⁽¹⁾⁽²⁾	R/W	0x0000	USB Receive Double Packet Buffer Disable	Section 22.6.44
0x342	USBTXDPKTBUFFDIS ⁽¹⁾⁽²⁾	R/W	0x0000	USB Transmit Double Packet Buffer Disable	Section 22.6.45
0x400	USBEP3 ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB External Power Control	Section 22.6.46

Table 22-3. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	Location
0x404	USBEPCCRIS ⁽¹⁾⁽²⁾	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	Section 22.6.47
0x408	USBEPCCIM ⁽²⁾⁽¹⁾	R/W	0x0000.0000	USB External Power Control Interrupt Mask	Section 22.6.48
0x40C	USBEPCCISC ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	Section 22.6.49
0x410	USBDRRIS ⁽¹⁾⁽²⁾	RO	0x0000.0000	USB Device RESUME Raw Interrupt Status	Section 22.6.50
0x414	USBDRIM ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB Device RESUME Interrupt Mask	Section 22.6.51
0x418	USBDRISC ⁽¹⁾⁽²⁾	W1C	0x0000.0000	USB Device RESUME Interrupt Status and Clear	Section 22.6.52
0x41C	USBGPCS ⁽¹⁾⁽²⁾	R/W	0x0000.0000	USB General-Purpose Control and Status	Section 22.6.53
0x450	USBDMASEL ⁽¹⁾⁽²⁾	R/W	0x0033.2211	USB DMA Select	Section 22.6.54

(1) This register is used in Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

(2) This register is used in Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

22.6 Register Descriptions

22.6.1 USB Device Functional Address Register (USBFADDR), offset 0x000

The USB function address 8-bit register (USBFADDR) contains the 7-bit address of the device part of the transaction.

When the USB controller is being used in device mode (the HOST bit in the USBDEVCTL register is clear), this register must be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

Mode(s): Device

For special considerations when writing this register, see the *Setting the Device Address* in [Section 22.3.1.1.4](#).

USBFADDR is shown in [Figure 22-3](#) and described in [Table 22-4](#).

Figure 22-3. Function Address Register (USBFADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-4. Function Address Register (USBFADDR) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	FUNCADDR	0-7Fh	Function Address of Device as received through SET_ADDRESS.

22.6.2 USB Power Management Register (USBPOWER), offset 0x001

The power management 8-bit register (USBPOWER) is used for controlling SUSPEND and RESUME signaling, and some basic operational aspects of the USB controller.

Mode(s): Host Device

USBPOWER in Host Mode is shown in [Figure 22-4](#) and described in [Table 22-5](#).

Figure 22-4. Power Management Register (USBPOWER) in Host Mode

7	4	3	2	1	0
Reserved		RESET	RESUME	SUSPEND	PWRDNPHY
R-0		R/W-0	R/W-0	R/W-1S	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-5. Power Management Register (USBPOWER) in Host Mode Field Descriptions

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
3	RESET	0	RESET signaling.
		0	Ends RESET signaling on the bus.
		1	Enables RESET signaling on the bus.
2	RESUME	0	RESUME signaling. The bit should be cleared by software 20 ms after being set.
		0	Ends RESUME signaling on the bus.
		1	Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND	0	SUSPEND mode
		0	No effect
		1	Enables SUSPEND mode.
0	PWRDNPHY	0	Power Down PHY
		0	No effect
		1	Powers down the internal USB PHY.

USBPOWER in Device Mode is shown in [Figure 22-5](#) and described in [Table 22-6](#).

Figure 22-5. Power Management Register (USBPOWER) in Device Mode

7	6	5	4	3	2	1	0
ISOUPDATE	SOFTCONN	Reserved		RESET	RESUME	SUSPEND	PWRDNPHY
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-6. Power Management Register (USBPOWER) in Device Mode Field Descriptions

Bit	Field	Value	Description
7	ISOUPDATE	0	Isochronous Update
		0	No effect
		1	The USB controller waits for an SOF token from the time the TXRDY bit is set in the USBTXCSRLn register before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.
6	SOFTCONN	0	Soft Connect/Disconnect
		0	The USB D+/D- lines are tri-stated.
		1	The USB D+/D- lines are enabled.
5-4	Reserved	0	Reserved

Table 22-6. Power Management Register (USBPOWER) in Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
3	RESET	0	RESET signaling Ends RESET signaling on the bus.
		1	Enables RESET signaling on the bus.
2	RESUME		RESUME signaling. The bit should be cleared by software 10 ms (a maximum of 15 ms) after being set.
		0	Ends RESUME signaling on the bus.
		1	Enables RESUME signaling when the Device is in SUSPEND mode.
1	SUSPEND		SUSPEND mode.
		0	This bit is cleared when software reads the interrupt register or sets the RESUME bit above.
		1	The USB controller is in SUSPEND mode.
0	PWRDNPHY		Power Down PHY
		0	No effect
		1	Powers down the internal USB PHY.

22.6.3 USB Transmit Interrupt Status Register (USBTXIS), offset 0x002

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB transmit interrupt status 16-bit read-only register (USBTXIS) indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–3. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP3 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

Mode(s): Host Device

USBTXIS is shown in [Figure 22-6](#) and described in [Table 22-7](#).

Figure 22-6. USB Transmit Interrupt Status Register (USBTXIS)

15																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset sho

Table 22-7. USB Transmit Interrupt Status Register (USBTXIS) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0 1	TX Endpoint 3 Interrupt No interrupt The Endpoint 3 transmit interrupt is asserted.
2	EP2	0 1	TX Endpoint 2 Interrupt No interrupt The Endpoint 2 transmit interrupt is asserted.
1	EP1	0 1	TX Endpoint 1 Interrupt No interrupt The Endpoint 1 transmit interrupt is asserted.
0	EP0	0 1	TX and RX Endpoint 0 Interrupt No interrupt The Endpoint 0 transmit and receive interrupt is asserted.

22.6.4 USB Receive Interrupt Status Register (USBRIXIS), offset 0x004

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB receive interrupt status 16-bit read-only register (USBRIXIS) indicates which interrupts are currently active for receive endpoints 1–3.

Note: Bits relating to endpoints that have not been configured always return 0. All active interrupts are cleared when this register is read.

Mode(s): Host Device

USBRIXIS is shown in [Figure 22-7](#) and described in [Table 22-8](#).

Figure 22-7. USB Receive Interrupt Status Register (USBRIXIS)

15																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-8. USB Receive Interrupt Status Register (USBRIXIS) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0 1	RX Endpoint 3 Interrupt No interrupt The Endpoint 3 receive interrupt is asserted.
2	EP2	0 1	RX Endpoint 2 Interrupt No interrupt The Endpoint 2 receive interrupt is asserted.
1	EP1	0 1	RX Endpoint 1 Interrupt No interrupt The Endpoint 1 receive interrupt is asserted.
0	Reserved	0	Reserved

22.6.5 USB Transmit Interrupt Enable Register (USBTXIE), offset 0x006

The USB transmit interrupt enable 16-bit register (USBTXIE) provides interrupt enable bits for the interrupts in the USBTXIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBTXIS register is set. When a bit is cleared, the interrupt in the USBTXIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

Mode(s): Host Device

USBTXIS is shown in [Figure 22-8](#) and described in [Table 22-9](#).

Figure 22-8. USB Transmit Interrupt Status Enable Register (USBTXIE)

15								4	3	2	1	0
Reserved									EP3	EP2	EP1	EP0
R-0									R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-9. USB Transmit Interrupt Status Register (USBTXIE) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0	TX Endpoint 3 Interrupt Enable The EP3 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP3 bit in the USBTXIS register is set.
2	EP2	0	TX Endpoint 2 Interrupt Enable The EP2 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP2 bit in the USBTXIS register is set.
1	EP1	0	TX Endpoint 1 Interrupt Enable The EP1 transmit interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP1 bit in the USBTXIS register is set.
0	EP0	0	TX and RX Endpoint 0 Interrupt Enable The EP0 transmit and receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP0 bit in the USBTXIS register is set.

22.6.6 USB Receive Interrupt Enable Register (USBRIE), offset 0x008

The USB receive interrupt enable 16-bit register (USBRIE) provides interrupt enable bits for the interrupts in the USBRIS register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the USBRIS register is set. When a bit is cleared, the interrupt in the USBRIS register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

Mode(s): Host Device

USBRIE is shown in [Figure 22-8](#) and described in [Table 22-9](#).

Figure 22-9. USB Receive Interrupt Enable Register (USBRIE)

15								4	3	2	1	0
Reserved									EP3	EP2	EP1	Rsvd
R-0									R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-10. USB Receive Interrupt Register (USBRIE) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0	RX Endpoint 3 Interrupt Enable The EP3 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP3 bit in the USBRIS register is set.
2	EP2	0	RX Endpoint 2 Interrupt Enable The EP2 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP2 bit in the USBRIS register is set.
1	EP1	0	RX Endpoint 1 Interrupt Enable The EP1 receive interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the EP1 bit in the USBRIS register is set.
0	Reserved	0	Reserved

22.6.7 USB General Interrupt Status Register (USBIS), offset 0x00A

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB general interrupt status 8-bit read-only register (USBIS) indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

Mode(s): Host Device

USBIS in Host Mode is shown in [Figure 22-10](#) and described in [Table 22-11](#).

Figure 22-10. USB General Interrupt Status Register (USBIS) in Host Mode

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-11. USB General Interrupt Status Register (USBIS) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	VBUSERR	0 1	VBUS Error No interrupt VBUS has dropped below the VBUS Valid threshold during a session.
6	SESREQ	0 1	Session Request No interrupt SESSION REQUEST signaling has been detected.
5	DISCON	0 1	Session Disconnect No interrupt A Device disconnect has been detected.
4	CONN	0 1	Session Connect No interrupt A Device connection has been detected.
3	SOF	0 1	Start of Frame No interrupt A new frame has started.
2	BABBLE	0 1	Babble Detected No interrupt Babble has been detected. This interrupt is active only after the first SOF has been sent.
1	RESUME	0 1	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. No effect RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	Reserved	0	Reserved

USBIS in Device Mode is shown in [Figure 22-11](#) and described in [Table 22-12](#).

Figure 22-11. USB General Interrupt Status Register (USBIS) in Device Mode

7	6	5	4	3	2	1	0
Reserved	DISCON	Reserved	SOF	RESET	RESUME	SUSPEND	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-12. USB General Interrupt Status Register (USBIS) in Device Mode Field Descriptions

Bit	Field	Value	Description
7-6	Reserved	0	Reserved
5	DISCON	0	Session Disconnect
		0	No interrupt
		1	The device has been disconnected from the host.
4	Reserved	0	Reserved
3	SOF	0	Start of frame
		0	No interrupt
		1	A new frame has started.
2	RESET	0	RESET Signaling Detected
		0	No interrupt
		1	RESET signaling has been detected on the bus.
1	RESUME	0	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used.
		0	No interrupt
		1	RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.
0	SUSPEND	0	SUSPEND Signaling Detected
		0	No interrupt
		1	SUSPEND signaling has been detected on the bus.

22.6.8 USB Interrupt Enable Register (USBIE), offset 0x00B

NOTE: Use caution when reading this register. Performing a read may change bit status.

The USB interrupt enable 8-bit register (USBIE) provides interrupt enable bits for each of the interrupts in USBIS. At reset interrupts 1 and 2 are enabled in device mode.

Mode(s): Host Device

USBIE in Host Mode is shown in [Figure 22-12](#) and described in [Table 22-13](#).

Figure 22-12. USB Interrupt Enable Register (USBIE) in Host Mode

7	6	5	4	3	2	1	0
VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	Reserved
R-W	R-W	R-W	R-W	R-W	R-W	R-W	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-13. USB Interrupt Enable Register (USBIE) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	VBUSERR	0 1	Enable VBUS Error Interrupt The VBUSERR interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set.
6	SESREQ	0 1	Enable Session Request The SESREQ interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the SESREQ bit in the USBIS register is set.
5	DISCON	0 1	Enable Disconnect Interrupt The DISCON interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	CONN	0 1	Enable Connect Interrupt The CONN interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set.
3	SOF	0 1	Start of Frame The SOF interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set.
2	BABBLE	0 1	Babble Detected The BABBLE interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set.
1	RESUME	0 1	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. The RESUME interrupt is suppressed and not sent to the interrupt controller. An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set.
0	Reserved	0	Reserved

USBIE in Device Mode is shown in [Figure 22-11](#) and described in [Table 22-12](#).

Figure 22-13. USB Interrupt Enable Register (USBIE) in Device Mode

7	6	5	4	3	2	1	0
Reserved	DISCON	Reserved	SOF	RESET	RESUME	SUSPEND	
R-0	R/W-0	R-0	R/W-0	R/W-1	RW-1	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-14. USB Interrupt Enable Register (USBIE) in Device Mode Field Descriptions

Bit	Field	Value	Description
7-6	Reserved	0	Reserved
5	DISCON	0	Enable Disconnect Interrupt The DISCON interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
4	Reserved	0	Reserved
3	SOF	0	Start of frame The SOF interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set.
2	RESET	0	RESET Signaling Detected The RESET interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the RESET bit in the USBIS register is set.
1	RESUME	0	RESUME Signaling Detected. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used. The RESUME interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set.
0	SUSPEND	0	SUSPEND Signaling Detected The SUSPEND interrupt is suppressed and not sent to the interrupt controller.
		1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.

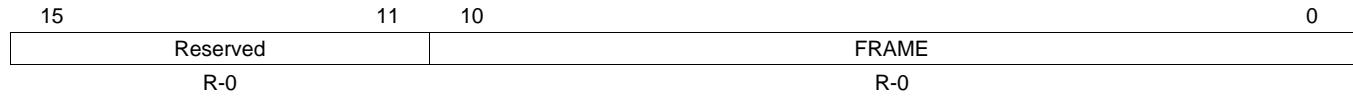
22.6.9 USB Frame Value Register (USBFRAME), offset 0x00C

The frame number 16-bit read-only register (USBFRAME) holds the last received frame number.

Mode(s): Host Device

USBFRAME is shown in [Figure 22-14](#) and described in [Table 22-15](#).

Figure 22-14. Frame Number Register (FRAME)



LEGEND: R = Read only; -n = value after reset

Table 22-15. Frame Number Register (FRAME) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	FRAME	0-7FFh	Last received frame number

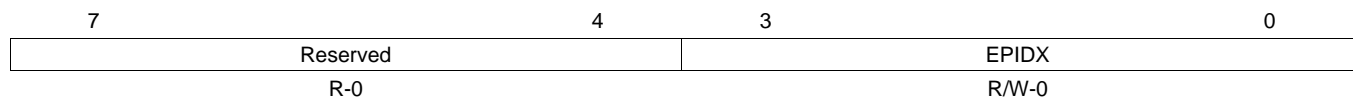
22.6.10 USB Endpoint Index Register (USBEPIDX), offset 0x00E

Each endpoint buffer can be accessed by configuring a FIFO size and starting address. The endpoint index 16-bit register (USBEPIDX) is used with the USBTXFIFOSZ, USBRXFIFOSZ, USBTXFIFOADD, and USBRXFIFOADD registers.

Mode(s): Host Device

USBEPIDX is shown in [Figure 22-15](#) and described in [Table 22-16](#).

Figure 22-15. USB Endpoint Index Register (USBEPIDX)



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-16. USB Endpoint Index Register (USBEPIDX) Field Descriptions

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
3-0	EPIDX	0-4h	Endpoint Index. This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15.

22.6.11 USB Test Mode Register (USBTEST), offset 0x00F

The USB test mode 8-bit register (USBTEST) is primarily used to put the USB controller into one of the four test modes for operation described in the USB Specification 2.0 , in response to a SET FEATURE: USBTESTMODE command. This register is not used in normal operation.

Note: Only one of these bits should be set at any time.

Mode(s): Host Device

USBTEST in Host Mode is shown in [Figure 22-16](#) and described in [Table 22-17](#).

Figure 22-16. USB Test Mode Register (USBTEST) in Host Mode

7	6	5	4	0
FORCEH	FIFOACC	FORCEFS	Reserved	
R/W-0	R/W1S-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

Table 22-17. USB Test Mode Register (USBTEST) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	FORCEH	0 1	Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit. No effect Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set.
6	FIFOACC	0 1	FIFO Access No effect Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.
5	FORCEFS	0 1	Force Full-Speed Mode The USB controller operates at Low Speed. Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
4-0	Reserved	0	Reserved

USBTEST in Device Mode is shown in [Figure 22-17](#) and described in [Table 22-18](#).

Figure 22-17. USB Test Mode Register (USBTEST) in Device Mode

7	6	5	4	0
Reserved	FIFOACC	FORCEFS	Reserved	
R-0	R/W1S-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

Table 22-18. USB Test Mode Register (USBTEST) in Device Mode Field Descriptions

Bit	Field	Value	Description
7	Reserved		Force Host Mode. While in this mode, status of the bus connection may be read using the DEV bit of the USBDEVCTL register. The operating speed is determined from the FORCEFS bit.
6	FIFOACC	0 1	FIFO Access No effect Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.

Table 22-18. USB Test Mode Register (USBTEST) in Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
5	FORCEFS		Force Full-Speed Mode
		0	The USB controller operates at Low Speed.
		1	Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
4-0	Reserved	0	Reserved

22.6.12 USB FIFO Endpoint n Register (USBFIFO[0]-USBFIFO[3])

NOTE: Use caution when reading these registers. Performing a read may change bit status.

The USB FIFO endpoint n 32-bit registers (USBFIFO[n]) provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

Transfers to and from FIFOs can be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering (see *Single-Packet Buffering* in [Section 22.3.1.1.2](#)). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

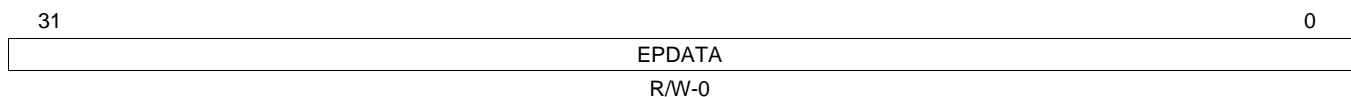
Following a STALL response or a transmit error on endpoint 1–3, the associated FIFO is completely flushed.

For the specific offset for each FIFO register see [Table 22-3](#).

Mode(s): Host Device

USBFIFO0-3 are shown in [Figure 22-18](#) and described in [Table 22-19](#).

Figure 22-18. USB FIFO Endpoint n Register (USBFIFO[n])



LEGEND: R/W = Read/Write; R = Read only; - n = value after reset

Table 22-19. USB FIFO Endpoint n Register (USBFIFO[n]) Field Descriptions

Bit	Field	Reset	Description
31-0	EPDATA	0x0000.0000	Endpoint Data. Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

22.6.13 USB Device Control Register (USBDEVCTL), offset 0x060

The USB device control 8-bit register (USBDEVCTL) is used for controlling and monitoring the USB V_{BUS} line. If the PHY is suspended, no PHY clock is received and the V_{BUS} is not sampled. In addition, in Host mode, USBDEVCTL provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

Mode(s):	Host	Device
----------	------	--------

USBDEVCTL is shown in [Figure 22-19](#) and described in [Table 22-20](#).

Figure 22-19. USB Device Control Register (USBDEVCTL)

7	6	5	4	3	2	1	0
DEV	FSDEV	LSDEV	VBUS		HOSTMODE	HOSTREQ	SESSION
R-1	R-0	R-0	R-0		R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-20. USB Device Control Register (USBDEVCTL) Field Descriptions

Bit	Field	Value	Description
7	DEV	0 1	Device mode The USB controller is operating on the OTG A side of the cable. The USB controller is operating on the OTG B side of the cable. Only valid while a session is in progress.
6	FSDEV	0 1	Full-Speed Device Detected A full-speed Device has not been detected on the port. A full-speed Device has been detected on the port.
5	LSDEV	0 1	Low-Speed Device Detected A low-speed Device has not been detected on the port. A low-speed Device has been detected on the port.
4-3	VBUS	0-3h 0 1h 2h 3h	These read-only bits encode the current VBus level as follows: Below Session End. VBUS is detected as under 0.5 V. Above Session End, below AValid. VBUS is detected as above 0.5 V and under 1.5 V. Above AValid, below VBusValid. VBUS is detected as above 1.5 V and below 4.75 V. Above VBusValid. VBUS is detected as above 4.75 V.
2	HOSTMODE	0 1	This read-only bit is set when the USB controller is acting as a Host. The USB controller is acting as a Device. The USB controller is acting as a Host. Only valid while a session is in progress.
1	HOSTREQ	0 1	When set, the USB controller will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed. No effect Initiates the Host Negotiation when SUSPENDmode is entered.

Table 22-20. USB Device Control Register (USBDEVCTL) Field Descriptions (continued)

Bit	Field	Value	Description
0	SESSION		Session Start/End When operating as a Host:
		0	When cleared by software, this bit ends a session.
		1	When set by software, this bit starts a session.
			When operating as a Device:
		0	The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.
		1	The USB controller has started a session. When set by software, the Session Request Protocol is initiated.
			Clearing this bit when the USB controller is not suspended results in undefined behavior.

22.6.14 USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ), offset 0x062

The USB transmit dynamic FIFO sizing 8-bit register (USBTXFIFOSZ) allows the selected TX endpoint FIFOs to be dynamically sized. USBEPIDX is used to configure each transmit endpoint's FIFO size.

Mode(s): Host Device

USBTXFIFOSZ is shown in [Figure 22-20](#) and described in [Table 22-21](#).

Figure 22-20. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-21. USB Transmit Dynamic FIFO Sizing Register (USBTXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB	0	Double Packet Buffering Support Single packet buffering is supported.
		1	Double packet buffering is enabled.
3-0	SZ	0h	Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes: 8
		1h	16
		2h	32
		3h	64
		4h	128
		5h	256
		6h	512
		7h	1024
		8h	2048
		9-Fh	Reserved

22.6.15 USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ), offset 0x063

The USB receive dynamic FIFO sizing 8-bit register (USBRXFIFOSZ) allows the selected RX endpoint FIFOs to be dynamically sized.

Mode(s): Host Device

USBRXFIFOSZ is shown in [Figure 22-21](#) and described in [Table 22-22](#).

Figure 22-21. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ)

7	5	4	3	0
Reserved		DPB	SZ	
R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-22. USB Receive Dynamic FIFO Sizing Register (USBRXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB	0	Double Packet Buffering Support Single packet buffering is supported.
		1	Double packet buffering is enabled.
3-0	SZ	0h	Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. Packet size in bytes: 8
		1h	16
		2h	32
		3h	64
		4h	128
		5h	256
		6h	512
		7h	1024
		8h	2048
		9-Fh	Reserved

22.6.16 USB Transmit FIFO Start Address Register (USBTXFIFOADD), offset 0x064

The USB transmit FIFO start address 16-bit register (USBTXFIFOADD) controls the start address of the selected transmit endpoint FIFOs.

Mode(s): Host Device

USBTXFIFOADDR is shown in [Figure 22-22](#) and described in [Table 22-23](#).

Figure 22-22. USB Transmit FIFO Start Address Register (USBTXFIFOADDR)

15	9	8	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-23. USB Transmit FIFO Start Address Register (USBTXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8-0	ADDR	0h	0
		1h	8
		2h	16
		3h	24
		4h	32
		5h	40
		6h	48
		7h	56
		8h	64
	
		1FFh	4095

22.6.17 USB Receive FIFO Start Address Register (USBRXFIFOADDR), offset 0x066

The USB receive FIFO start address 16-bit register (USBRXFIFOADDR) controls the start address of the selected receive endpoint FIFOs.

Mode(s): Host Device

USBRXFIFOADDR is shown in [Figure 22-23](#) and described in [Table 22-24](#).

Figure 22-23. USB Receive FIFO Start Address Register (USBRXFIFOADDR)

15	9	8	0
Reserved		ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-24. USB Receive FIFO Start Address Register (USBRXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8-0	ADDR		Start Address of the endpoint FIFO in units of 8 bytes.
		0h	0
		1h	8
		2h	16
		3h	24
		4h	32
		5h	40
		6h	48
		7h	56
		8h	64
	
		1FFh	4095

22.6.18 USB Connect Timing Register (USBCONTIM), offset 0x07A

The USB connect timing 8-bit configuration register (USBCONTIM) specifies connection and negotiation delays.

Mode(s): Host Device

USBCONTIM is shown in [Figure 22-24](#) and described in [Table 22-25](#).

Figure 22-24. USB Connect Timing Register (USBCONTIM)

7	4	3	0
WTCN		WTID	
R/W-1		R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-25. USB Connect Timing Register (USBCONTIM) Field Descriptions

Bit	Field	Value	Description
7-4	WTCN	5h	The connect wait field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 μ s.
3-0	WTID	Ch	The wait ID field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms.

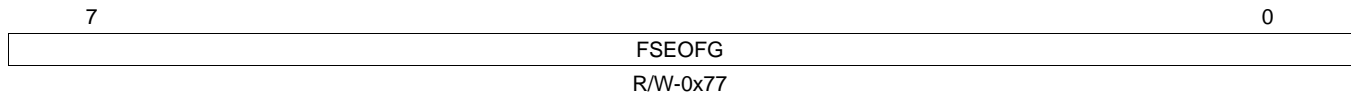
22.6.19 USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF), offset 0x07D

USB full-speed last transaction to end of frame timing 8-bit configuration register (USBFSEOF) specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

Mode(s): Host Device

USBFSEOF is shown in [Figure 22-25](#) and described in [Table 22-26](#).

Figure 22-25. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF)



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-26. USB Full-Speed Last Transaction to End of Frame Timing Register (USBFSEOF) Field Descriptions

Bit	Field	Reset	Description
7-0	FSEOFG	77h	The full-speed end-of-frame gap field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 μ s.

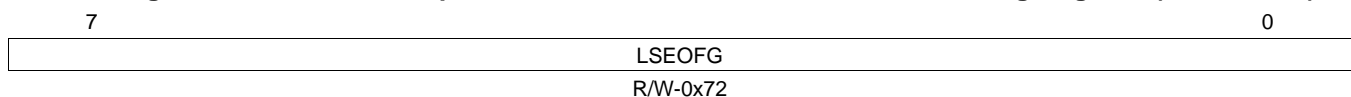
22.6.20 USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF), offset 0x07E

The USB low-speed last transaction to end of frame timing 8-bit configuration register (USBLSEOF) specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

Mode(s): Host Device

USBLSEOF is shown in [Figure 22-26](#) and described in [Table 22-27](#).

Figure 22-26. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF)



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-27. USB Low-Speed Last Transaction to End of Frame Timing Register (USBLSEOF) Field Descriptions

Bit	Field	Reset	Description
7-0	LSEOFG	72h	The low-speed end-of-frame gap field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 μ s. The default corresponds to 121.6 μ s.

22.6.21 USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[0]-USBTXFUNCADDR[3])

The transmit functional address endpoint n 8-bit registers (USBTXFUNCADDR[n]) record the address of the target function to be accessed through the associated endpoint (EP n). USBTXFUNCADDR x must be defined for each transmit endpoint that is used.

Note: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBTXFUNCADDR[n] registers are shown in [Figure 22-27](#) and described in [Table 22-28](#).

Figure 22-27. USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[n])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

Table 22-28. USB Transmit Functional Address Endpoint n Registers (USBTXFUNCADDR[n]) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

22.6.22 USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[0]-USBTXHUBADDR[3])

The transmit hub address endpoint n 8-bit read/write registers (USBTXHUBADDR[n]), like USBTXHUBPORT[n], must be written only when a USB device is connected to transmit endpoint EP n via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBTXHUBADDR[n] registers are shown in [Figure 22-27](#) and described in [Table 22-28](#).

Figure 22-28. USB Transmit Hub Address Endpoint n Registers (USBTXHUBADDR[n])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-29. USB Transmit Hub Address Endpoint n Registers(USBTXHUBADDR[n])
Field Descriptions**

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

22.6.23 USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[0]-USBTXHUBPORT[3])

The transmit hub port endpoint n 8-bit read/write registers (USBTXHUBPORT[n]), like USBTXHUBADDR[n], must be written only when a full- or low-speed Device is connected to transmit endpoint EP n via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBTXHUBPORT n registers are shown in [Figure 22-29](#) and described in [Table 22-30](#).

Figure 22-29. USB Transmit Hub Port Endpoint n Registers (USBTXHUBPORT[n])

7	6	0
Reserved	PORT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-30. USB Transmit Hub Port Endpoint n Registers(USBTXHUBPORT[n])
Field Descriptions**

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	PORT	0	Hub Port specifies the USB hub port number.

22.6.24 USB Receive Functional Address Endpoint n Registers (USBXFUNCADDR[1]-USBXFUNCADDR[3])

The receive functional address endpoint n 8-bit read/write registers (USBXFUNCADDR[n]) record the address of the target function to be accessed through the associated endpoint (EP n). USBXFUNCADDR x must be defined for each receive endpoint that is used.

Note: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBXFUNCADDR[n] registers are shown in [Figure 22-30](#) and described in [Table 22-31](#).

Figure 22-30. USB Receive Functional Address Endpoint n Registers (USBFIFO[n])

7	6	0
Reserved	ADDR	
R-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-31. USB Receive Functional Address Endpoint n Registers(USBFIFO[n])
Field Descriptions**

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

22.6.25 USB Receive Hub Address Endpoint n Registers (USBXHUBADDR[1]-USBXHUBADDR[3])

The receive hub address endpoint n 8-bit read/write registers (USBXHUBADDR[n]), like [n], must be written only when a full- or low-speed Device is connected to receive endpoint EP n via a USB 2.0 hub. Each register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBXHUBADDR[n] registers are shown in [Figure 22-31](#) and described in [Table 22-32](#).

Figure 22-31. USB Receive Hub Address Endpoint n Registers (USBXHUBADDR[n])

7	6	0
MULTTRAN	ADDR	
R/w-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-32. USB Receive Hub Address Endpoint n Registers(USBXHUBADDR[n])
Field Descriptions**

Bit	Field	Value	Description
7	MULTTRAN	0	Multiple Translators Clear to indicate that the hub has a single transaction translator.
		1	Set to indicate that the hub has multiple transaction translators.
6-0	ADDR	0	Device Address specifies the USB bus address for the target Device.

22.6.26 USB Receive Hub Port Endpoint n Registers (USBXHUBPORT[1]-USBXHUBPORT[3])

The receive hub port endpoint n 8-bit read/write registers (USBXHUBPORT[n]), like USBXHUBADDR[n], must be written only when a full- or low-speed device is connected to receive endpoint EP n via a USB 2.0 hub. Each register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

Note: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBXHUBPORT n registers are shown in [Figure 22-32](#) and described in [Table 22-33](#).

Figure 22-32. USB Transmit Hub Port Endpoint n Registers (USBXHUBPORT[n])

7	6	0
Reserved	PORT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-33. USB Transmit Hub Port Endpoint n Registers(USBXHUBPORT[n])
Field Descriptions**

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	PORT	0	Hub Port specifies the USB hub port number.

22.6.27 USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[1]-USBTXMAXP[3])

The USB maximum transmit data endpoint n 16-bit registers (USBTXMAXP[n]) define the maximum amount of data that can be transferred through the selected transmit endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the FLUSH bit in USBTXCSRLn) after writing the new value to this register.

Note: USBTXMAXP[n] must be set to an even number of bytes for proper interrupt generation in DMA Basic Mode.

For the specific offset for each register see [Table 22-3](#).

Mode(s):	Host	Device
-----------------	------	--------

The USBTXMAXP[n] registers are shown in [Figure 22-33](#) and described in [Table 22-34](#).

Figure 22-33. USB Maximum Transmit Data Endpoint n Registers (USBTXMAXP[n])

15	11	10	0
Reserved		MAXLOAD	
R-0		R/W-000	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-34. USB Maximum Transmit Data Endpoint n Registers(USBTXMAXP[n])
Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXLOAD		Maximum Payload specifies the maximum payload in bytes per transaction.

22.6.28 USB Control and Status Endpoint 0 Low Register (USBCSRL0), offset 0x102

The USB control and status endpoint 0 low 8-bit register (USBCSRL0) provides control and status bits for endpoint 0.

Mode(s): Host Device

USBCSRL0 in Host mode is shown in [Figure 22-34](#) and described in [Table 22-35](#).

Figure 22-34. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Host Mode

7	6	5	4	3	2	1	0
NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-35. USB Control and Status Endpoint 0 Low Register(USBCSRL0) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO	0 1	NAK Timeout. Software must clear this bit to allow the endpoint to continue. No timeout Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register.
6	STATUS	0 1	Status Packet. Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction. No transaction Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set. This bit is automatically cleared when the STATUS stage is over.
5	REQPKT	0 1	Request Packet. This bit is cleared when the RXRDY bit is set. No request Requests an IN transaction.
4	ERROR	0 1	Error. Software must clear this bit. No error Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the USBTXIS register is also set in this situation.
3	SETUP	0 1	Setup Packet. Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet. Sends an OUT token. Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
2	STALLED	0 1	Endpoint Stalled. Software must clear this bit. No handshake has been received. A STALL handshake has been received.
1	TXRDY	0 1	Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent. No transmit packet is ready. Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.
0	RXRDY	0 1	Receive Packet Ready. Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO. No receive packet has been received. Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.

USBCSRL0 in Device mode is shown in [Figure 22-35](#) and described in [Table 22-36](#).

Figure 22-35. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode

7	6	5	4	3	2	1	0
SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
W1C-0	W1C-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-36. USB Control and Status Endpoint 0 Low Register (USBCSRL0) in Device Mode Field Descriptions

Bit	Field	Value	Description
7	SETENDC	0 1	Setup End Clear No effect Writing a 1 to this bit clears the SETEND bit.
6	RXRDYC	0 1	RXRDY Clear No effect Writing a 1 to this bit clears the RXRDY bit.
5	STALL	0 1	Send Stall. No effect Terminates the current transaction and transmits the STALL handshake. This bit is cleared automatically after the STALL handshake is transmitted.
4	SETEND	0 1	Setup end. A control transaction has not ended or ended after the DATAEND bit was set. A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the SETENDC bit.
3	DATAEND	0 1	Data end. No effect Set this bit in the following situations: <ul style="list-style-type: none"> When setting TXRDY for the last data packet When clearing RXRDY after unloading the last data packet When setting TXRDY for a zero-length data packet This bit is cleared automatically.
2	STALLED	0 1	Endpoint Stalled. Software must clear this bit. A STALL handshake has not been transmitted. A STALL handshake has been transmitted.
1	TXRDY	0 1	Transmit Packet Ready. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent. No transmit packet is ready. Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.
0	RXRDY	0 1	Receive Packet Ready. No receive packet has been received. A data packet has been received. The EP0 bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the RXRDYC bit.

22.6.29 USB Control and Status Endpoint 0 High Register (USBCSRH0), offset 0x103

The USB control and status endpoint 0 high 8-bit register (USBCSRH0) provides control and status bits for endpoint 0.

Mode(s): Host Device

USBCSRH0 in Host mode is shown in [Figure 22-36](#) and described in [Table 22-37](#).

Figure 22-36. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode

7	3	2	1	0
Reserved		DTWE	DT	FLUSH
R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Host Mode Field Descriptions

Bit	Field	Value	Description
7-3	Reserved	0	Reserved
2	DTWE	0	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. The DT bit cannot be written.
		1	Enables the current state of the endpoint 0 data toggle to be written (see DT bit).
1	DT		Data Toggle. When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0.
0	FLUSH	0	Flush FIFO. This bit is automatically cleared after the flush is performed. No effect
		1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. Note: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

USBCSRH0 in Device mode is shown in [Figure 22-37](#) and described in [Table 22-38](#).

Figure 22-37. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode

7	1	0
Reserved		FLUSH
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-38. USB Control and Status Endpoint 0 High Register (USBCSRH0) in Device Mode Field Descriptions

Bit	Field	Value	Description
7-1	Reserved	0	Reserved
0	FLUSH	0	Flush FIFO. This bit is automatically cleared after the flush is performed. No effect
		1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. Note: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

22.6.30 USB Receive Byte Count Endpoint 0 Register (USBCOUNT0), offset 0x108

The USB receive byte count endpoint 0 8-bit read-only register (USBCOUNT0) indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the RXRDY bit is set.

Mode(s): Host Device

USBCOUNT0 is shown in [Figure 22-38](#) and described in [Table 22-28](#).

Figure 22-38. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0)

7	6	0
Reserved	COUNT	
R-0	R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-39. USB Receive Byte Count Endpoint 0 Register (USBCOUNT0) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	COUNT	0	FIFO Count. COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

22.6.31 USB Type Endpoint 0 Register (USBTYPE0), offset 0x10A

The USB type endpoint 0 8-bit register (USBTYPE0) must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

Mode(s): Host

USBTYPE0 is shown in [Figure 22-39](#) and described in [Table 22-40](#).

Figure 22-39. USB Type Endpoint 0 Register (USBTYPE0)

7	6	5	0
SPEED		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-40. USB Type Endpoint 0 Register (USBTYPE0) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED	0	Operating Speed specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller.
		0-1h	Reserved
		2h	Full
		3h	Low
5-0	Reserved	0	Reserved

22.6.32 USB NAK Limit Register (USBNAKLMT), offset 0x10B

The USB NAK limit 8-bit read-only register (USBNAKLMT) sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their USBTXINTERVAL[n] and USBRXINTERVAL[n] registers.)

The number of frames selected is $2(m-1)$ (where m is the value set in the register, with valid values of 2–16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

Note: A value of 0 or 1 disables the NAK timeout function.

Mode(s): Host

USBNAKLMT is shown in [Figure 22-40](#) and described in [Table 22-41](#).

Figure 22-40. USB NAK Limit Register (USBNAKLMT)

7	5	4	0
Reserved		NAKLMT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-41. USB NAK Limit Register (USBNAKLMT) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4-0	NAKLMT	0	EP0 NAK Limit specifies the number of frames after receiving a stream of NAK responses.

22.6.33 USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[1]-USBTXCSRL[3])

The USB transmit control and status endpoint n low 8-bit registers (USBTXCSRL[n]) provide control and status bits for transfers through the currently selected transmit endpoint.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBTXCSRL[n] registers in Host Mode are shown in [Figure 22-41](#) and described in [Table 22-42](#).

Figure 22-41. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n]) in Host Mode

7	6	5	4	3	2	1	0
NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; - n = value after reset

Table 22-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n]) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO	0 1	NAK Timeout. Software must clear this bit to allow the endpoint to continue. No timeout Bulk endpoints only: Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBTXINTERVAL[n] register.
6	CLRDT	0 1	Clear DataToggle No effect Writing a 1 to this bit clears the DT bit in the USBTXCSRH[n] register.
5	STALLED	0 1	Endpoint Stalled. Software must clear this bit. A STALL handshake has not been received Indicates that a STALL handshake has been received. When this bit is set, any DMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared.
4	SETUP	0 1	Setup Packet. No SETUP token is sent. Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set. Note: Setting this bit also clears the DT bit in the USBTXCSRH[n] register.
3	FLUSH	0 1	Flush FIFO. This bit can be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. No effect Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation. Note: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.
2	ERROR	0 1	Error. Software must clear this bit. No error Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation. Note: This bit is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	FIFONE	0 1	FIFO Not Empty The FIFO is empty At least one packet is in the transmit FIFO.

**Table 22-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n])
in Host Mode Field Descriptions (continued)**

Bit	Field	Value	Description
0	TXRDY		Transmit Packet Ready. This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.
		0	No transmit packet is ready.
		1	Software sets this bit after loading a data packet into the TX FIFO.

The USBTXCSRL[n] registers in Device Mode are shown in [Table 22-42](#) and described in [Figure 22-42](#).

**Figure 22-42. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n])
in Device Mode**

7	6	5	4	3	2	1	0
Reserved	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-43. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n])
in Device Mode Field Descriptions**

Bit	Field	Value	Description
7	Reserved	0	Reserved
6	CLRDT	0	Clear Data Toggle No effect
		1	Writing a 1 to this bit clears the DT bit in the USBTXCSRH[n] register.
5	STALLED	0	Endpoint Stalled. Software must clear this bit. A STALL handshake has not been transmitted.
		1	A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared.
4	STALL	0	Send Stall. Software clears this bit to terminate the STALL condition. Note: This bit has no effect in isochronous transfers. No effect
		1	Issues a STALL handshake to an IN token.
3	FLUSH	0	Flush FIFO. This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. Note: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted. No effect
		1	Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation. This bit is cleared automatically.
2	UNDRN	0	Underrun. Software must clear this bit. No underrun
		1	An IN token has been received when TXRDY is not set.
1	FIFONE	0	FIFO Not Empty The FIFO is empty.
		1	At least one packet is in the transmit FIFO.

Table 22-43. USB Transmit Control and Status Endpoint n Low Register (USBTXCSRL[n])
in Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
0	TXRDY		Transmit Packet Ready. This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.
		0	No transmit packet is ready.
		1	Software sets this bit after loading a data packet into the TX FIFO.
			This bit is cleared by writing a 1 to the RXRDYC bit.

22.6.34 USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[1]-USBTXCSRH[3])

The USB transmit control and status endpoint n high 8-bit registers (USBTXCSRH[n]) provide additional control for transfers through the currently selected transmit endpoint.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBTXCSRH[n] registers in Host Mode are shown in [Figure 22-43](#) and described in [Table 22-44](#).

Figure 22-43. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n]) in Host Mode

7	6	5	4	3	2	1	0
AUTOSET	Reserved	MODE	DMAEN	FDT	DMAMOD	DTWE	DT
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; - n = value after reset

Table 22-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n]) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOSET	0 1	Auto Set The TXRDY bit must be set manually. Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXP[n]) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
6	Reserved	0	Reserved
5	MODE	0 1	Mode Note: This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions. Enables the endpoint direction as RX. Enables the endpoint direction as TX.
4	DMAEN	0 1	DMA Request Enable Note: Three TX and three /RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMAXTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. Disables the DMA request for the transmit endpoint. Enables the DMA request for the transmit endpoint.
3	FDT	0 1	Force Data Toggle No effect Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints. Note: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.
2	DMAMOD	0 1	DMA Request Mode Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. An interrupt is generated after every DMA packet transfer. An interrupt is generated only after the entire DMA transfer is complete. Note: This bit is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	DTWE	0 1	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. The DT bit cannot be written. Enables the current state of the transmit endpoint data to be written (see DT bit).

**Table 22-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n])
in Host Mode Field Descriptions (continued)**

Bit	Field	Value	Description
0	DT		Data Toggle. When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit can be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.

The USBTXCSRH[n] registers in Device Mode are shown in [Figure 22-44](#) and described in [Table 22-45](#).

**Figure 22-44. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n])
in Device Mode**

7	6	5	4	3	2	1	0
AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	Reserved	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-45. USB Transmit Control and Status Endpoint n High Register (USBTXCSRH[n])
in Device Mode Field Descriptions**

Bit	Field	Value	Description
7	AUTOSET	0	Auto Set The TXRDY bit must be set manually.
		1	Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXP[n]) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
6	ISO	0	Isochronous Transfers Enables the transmit endpoint for bulk or interrupt transfers.
		1	Enables the transmit endpoint for isochronous transfers.
5	MODE		Mode Note: This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
		0	Enables the endpoint direction as RX.
		1	Enables the endpoint direction as TX.
4	DMAEN		DMA Request Enable Note: Three TX and three /RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.
		0	Disables the DMA request for the transmit endpoint.
		1	Enables the DMA request for the transmit endpoint.
3	FDT		Force Data Toggle
		0	No effect
		1	Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD		DMA Request Mode Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
		0	An interrupt is generated after every DMA packet transfer.
		1	An interrupt is generated only after the entire DMA transfer is complete.
0	Reserved	0	Reserved

22.6.35 USB Maximum Receive Data Endpoint n Registers (USBXMAXP[1]-USBXMAXP[3])

The USB maximum receive data endpoint n 16-bit registers (USBXMAXP[n]) define the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

Note: USBXMAXP[n] must be set to an even number of bytes for proper interrupt generation in DMA Basic Mode.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBXMAXP[n] registers are shown in [Figure 22-45](#) and described in [Table 22-46](#).

Figure 22-45. USB Maximum Receive Data Endpoint n Registers (USBXMAXP[n])

15	11	10	0
Reserved		MAXLOAD	
R-0		R/W-000	

LEGEND: R/W = Read/Write; R = Read only; - n = value after reset

Table 22-46. USB Maximum Receive Data Endpoint n Registers (USBTXMAXP[n]) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXLOAD		Maximum Payload specifies the maximum payload in bytes per transaction.

22.6.36 USB Receive Control and Status Endpoint n Low Register (USBRXCSRL[1]-USBRXCSRL[3])

The USB receive control and status endpoint n low 8-bit register (USBCSRL[n]) provides control and status bits for transfers through the currently selected receive endpoint.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBCSRL[n] registers in Host mode are shown in [Figure 22-46](#) and described in [Table 22-47](#).

Figure 22-46. USB Receive Control and Status Endpoint n Low Register (USBCSRL[n]) in Host Mode

7	6	5	4	3	2	1	0
CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
W1C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; - n = value after reset

Table 22-47. USB Control and Status Endpoint n Low Register(USBCSRL[n]) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	NAKTO	0 1	Clear Data Toggle. No effect Writing a 1 to this bit clears the DT bit in the USBRXCSRH[n] register.
6	STALLED	0 1	Endpoint Stalled. Software must clear this bit. No handshake has been received. A STALL handshake has been received. The EP n bit in the USBRXIS register is also set.
5	REQPKT	0 1	Request Packet. This bit is cleared when the RXRDY bit is set. No request Requests an IN transaction.
4	FLUSH	0 1	Flush FIFO. If the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. Note: This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted. No effect Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.
3	DATAERR / NAKTO	0 1	Data Error / NAK Timeout Normal operation Isochronous endpoints only: Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. Bulk endpoints only: Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVAL[n] register. Software must clear this bit to allow the endpoint to continue.
2	ERROR	0 1	Error. Software must clear this bit. Note: This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero. No error Three attempts have been made to receive a packet and no data packet has been received. The EP n bit in the USBRXIS register is set in this situation.
1	FULL	0 1	FIFO Full The receive FIFO is not full. No more packets can be loaded into the receive FIFO.

**Table 22-47. USB Control and Status Endpoint n Low Register(USBCSRL[n])
in Host Mode Field Descriptions (continued)**

Bit	Field	Value	Description
0	RXRDY		Receive Packet Ready. If the AUTOCLR bit in the USBRXCSRH[n] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[n] bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.
		0	No data packet has been received.
		1	Indicates that a data packet has been received. The EP n bit in the USBTXIS register is also set in this situation.

USBCSRL0 in Device mode is shown in [Figure 22-47](#) and described in [Table 22-48](#).

**Figure 22-47. USB Control and Status Endpoint n Low Register (USBCSRL[n])
in Device Mode**

7	6	5	4	3	2	1	0
CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
W1C-0	W1C-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-48. USB Control and Status Endpoint 0 Low Register(USBCSRL[n])
in Device Mode Field Descriptions**

Bit	Field	Value	Description
7	CLRDT		Clear Data Toggle
		0	No effect
		1	Writing a 1 to this bit clears the DT bit in the USBRXCSRH[n] register.
6	STALLED		Endpoint Stalled. Software must clear this bit.
		0	A STALL handshake has been transmitted.
		1	A STALL handshake has been transmitted.
5	STALL		Send Stall. Software must clear this bit to terminate the STALL condition. Note: This bit has no effect where the endpoint is being used for isochronous transfers.
		0	No effect
		1	Issues a STALL handshake.
4	FLUSH		Flush FIFO. The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.
		0	No effect
		1	Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note: This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.
3	DATAEND		Data error. This bit is cleared when RXRDY is cleared. Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
		0	Normal operation
		1	Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared automatically.
2	OVER		Overrun. Software must clear this bit. Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.
		0	No overrun error
		1	Indicates an OUT packet cannot be loaded into the receive FIFO.

Table 22-48. USB Control and Status Endpoint 0 Low Register(USBCSRL[n])
in Device Mode Field Descriptions (continued)

Bit	Field	Value	Description
1	FULL	0	FIFO Full The receive FIFO is not full.
		1	No more packets can be loaded into the receive FIFO.
0	RXRDY		Receive Packet Ready. If the AUTOCLR bit in the USBRXCSRH[n] register is set, then the this bit is automatically cleared when a packet of USBRXMAXP[n] bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.
		0	No data packet has been received.
		1	A data packet has been received. The EPn bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the RXRDYC bit.

22.6.37 USB Receive Control and Status Endpoint n High Register (USBXCSRH[1]-USBXCSRH[3])

The USB receive control and status endpoint n high 8-bit register (USBCSRL[n]) provides additional control and status bits for transfers through the currently selected receive endpoint.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBCSRH[n] registers in OTG A/Host mode are shown in [Figure 22-48](#) and described in [Table 22-49](#).

Figure 22-48. USB Receive Control and Status Endpoint n High Register (USBCSRH[n]) in Host Mode

7	6	5	4	3	2	1	0
AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	Reserved
W1C-0	R/W-0	R/W-0	R-0	R/W-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; - n = value after reset

Table 22-49. USB Control and Status Endpoint n High Register (USBCSRH[n]) in Host Mode Field Descriptions

Bit	Field	Value	Description
7	AUTOCL	0 1	Auto Clear No effect Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[n] bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using DMA to unload the receive FIFO as data is read from the receive FIFO in 4-byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXP[n] register, see Section 22.3.3 .
6	AUTORQ	0 1	Auto Request Note: This bit is automatically cleared when a short packet is received. No effect Enables the REQPKT bit to be automatically set when the RXRDY bit is cleared.
5	DMAEN	0 1	DMA Request Enable Note: Three TX and three RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. Disables the DMA request for the receive endpoint. Enables the DMA request for the receive endpoint.
4	PIDERR	0 1	PID Error. This bit is ignored in bulk or interrupt transactions. No error Indicates a PID error in the received packet of an isochronous transaction.
3	DMAMOD	0 1	DMAMOD Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. An interrupt is generated after every DMA packet transfer. An interrupt is generated only after the entire DMA transfer is complete.
2	DTWE	0 1	Data Toggle Write Enable. This bit is automatically cleared once the new value is written. The DT bit cannot be written. Enables the current state of the receive endpoint data to be written (see DT bit).
1	DT		Data Toggle. When read, this bit indicates the current state of the receive data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.
0	Reserved	0	Reserved

The USBCSRH[n] registers in Device mode are shown in [Figure 22-49](#) and described in [Table 22-50](#).

Figure 22-49. USB Control and Status Endpoint *n* High Register (USBCSRH[n]) in Device Mode

7	6	5	4	3	2	0
AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD		Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R-0

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 22-50. USB Control and Status Endpoint 0 High Register(USBCSRH[n])
in Device Mode Field Descriptions**

Bit	Field	Value	Description
7	AUTOCL	0 1	Auto Clear No effect Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXP[n] bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using DMA to unload the receive FIFO as data is read from the receive FIFO in 4-byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXP[n] register, see Section 22.3.3 .
6	ISO	0 1	Isochronous Transfers Enables the receive endpoint for isochronous transfers. Enables the receive endpoint for bulk/interrupt transfers.
5	DMAEN	0 1	DMA Request Enable Note: Three TX and three RX endpoints can be connected to the DMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. Disables the DMA request for the receive endpoint. Enables the DMA request for the receive endpoint.
4	DISNYET/PI DERR	0 1	Disable NYET / PID Error No effect For bulk or interrupt transactions: Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full. For isochronous transactions: Indicates a PID error in the received packet.
3	DMAMOD	0 1	DMA Request Mode Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared. An interrupt is generated after every DMA packet transfer. An interrupt is generated only after the entire DMA transfer is complete.
0	Reserved	0	Reserved

22.6.38 USB Receive Byte Count Endpoint n Registers (USBXCOUNT[1]-USBXCOUNT[3])

The USB receive byte count endpoint n 16-bit read-only registers hold the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

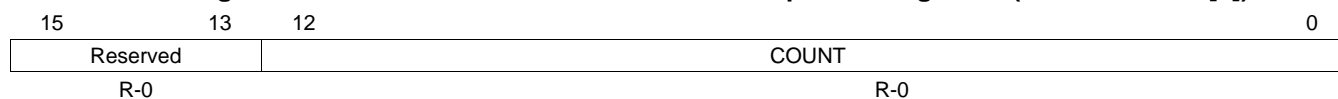
Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the USBXCSRLn register is set.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host Device

The USBXCOUNT[n] registers are shown in [Figure 22-50](#) and described in [Table 22-51](#).

Figure 22-50. USB Maximum Receive Data Endpoint n Registers (USBXCOUNT[n])



LEGEND: R/W = Read/Write; R = Read only; - n = value after reset

**Table 22-51. USB Maximum Receive Data Endpoint n Registers (USBXCOUNT[n])
Field Descriptions**

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	COUNT		Receive Packet Count indicates the number of bytes in the receive packet.

22.6.39 USB Host Transmit Configure Type Endpoint *n* Register (USBTXTYPE[1]-USBTXTYPE[3])

The USB host transmit configure type endpoint *n* 8-bit registers (USBTXTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBTXTYPE[*n*] registers are shown in [Figure 22-51](#) and described in [Table 22-52](#).

Figure 22-51. USB Host Transmit Configure Type Endpoint *n* Register (USBTXTYPE[*n*])

7	6	5	4	3	0
SPEED		PROTO		TEP	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 22-52. USB Host Transmit Configure Type Endpoint *n* Register(USBTXTYPE[*n*])
Field Descriptions**

Bit	Field	Value	Description
7-6	SPEED	0h	Operating Speed. This bit field specifies the operating speed of the target Device: Default. The target is assumed to be using the same connection speed as the USB controller.
		1h	Reserved
		2h	Full
		3h	Low
5-4	PROTO	0h	Protocol. Software must configure this bit field to select the required protocol for the transmit endpoint: Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	TEP	0	Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

22.6.40 USB Host Transmit Interval Endpoint *n* Register (USBTXINTERVAL[1]USBTXINTERVAL[3])

The USB host transmit interval endpoint *n* 8-bit registers (USBTXINTERVAL[*n*]), for interrupt and isochronous transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVAL[*n*] registers values define a number of frames, as follows:

Table 22-53. USBTXINTERVAL[*n*] Frame Numbers

Transfer Type	Speed	Valid Values (<i>m</i>)	Interpretation
Interrupt	Low-speed or Full-speed	0x01-0xFF	The polling interval is <i>m</i> frames.
Isochronous	Full-speed	0x01-0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-speed	0x02-0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBTXINTERVAL[*n*] registers are shown in [Figure 22-51](#) and described in [Table 22-52](#).

Figure 22-52. USB Host Transmit Interval Endpoint *n* Register (USBTXINTERVAL[*n*])

7	TXPOLL / NAKLMT	0
R/W-0		

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 22-54. USB Host Transmit Interval Endpoint *n* Register(USBTXINTERVAL[*n*])
Field Descriptions**

Bit	Field	Value	Description
7-0	TXPOLL / NAKLMT	0	TX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See Table 22-53 for valid entries; other values are reserved.

22.6.41 USB Host Configure Receive Type Endpoint *n* Register (USBRTYPE[1]-USBRTYPE[3])

The USB host configure receive type endpoint *n* 8-bit registers (USBRTYPE[*n*]) must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBRTYPE[*n*] registers are shown in [Figure 22-53](#) and described in [Table 22-55](#).

Figure 22-53. USB Host Configure Receive Type Endpoint *n* Register (USBRTYPE[*n*])

7	6	5	4	3	0
SPEED		PROTO		TEP	
R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 22-55. USB Host Configure Receive Type Endpoint *n* Register(USBRTYPE[*n*])
Field Descriptions**

Bit	Field	Value	Description
7-6	SPEED	0h 1h 2h 3h	Operating Speed. This bit field specifies the operating speed of the target Device: Default. The target is assumed to be using the same connection speed as the USB controller. Reserved Full Low
5-4	PROTO	0h 1h 2h 3h	Protocol. Software must configure this bit field to select the required protocol for the receive endpoint: Control Isochronous Bulk Interrupt
3-0	TEP	0	Target Endpoint Number. Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.

22.6.42 USB Host Receive Polling Interval Endpoint n Register (USBXINTERVAL[1]-USBXINTERVAL[3])

The USB host receive polling interval endpoint n 8-bit registers (USBXINTERVAL[n]), for interrupt and isochronous transfers, define the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBXINTERVAL[n] registers values define a number of frames, as follows:

Table 22-56. USBXINTERVAL[n] Frame Numbers

Transfer Type	Speed	Valid Values (m)	Interpretation
Interrupt	Low-speed or Full-speed	0x01-0xFF	The polling interval is m frames.
Isochronous	Full-speed	0x01-0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-speed	0x02-0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBXINTERVAL[n] registers are shown in [Figure 22-51](#) and described in [Table 22-52](#).

Figure 22-54. USB Host Receive Polling Interval Endpoint n Register (USBXINTERVAL[n])

7	TXPOLL / NAKLMT	0
R/W-0		

LEGEND: R/W = Read/Write; - n = value after reset

**Table 22-57. USB Host Receive Polling Interval Endpoint n Register(USBXINTERVAL[n])
Field Descriptions**

Bit	Field	Value	Description
7-0	TXPOLL / NAKLMT	0	TX Polling / NAK Limit The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See Table 22-56 for valid entries; other values are reserved.

22.6.43 USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[1]-USBRQPKTCOUNT[3])

The USB receive packet count in block transfer endpoint n 16-bit read/writer registers are used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint n . The USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the USBRXCSRH[n] register has been set. For more information about IN transactions as a host, see [Section 22.3.2.2](#).

Note: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

For the specific offset for each register see [Table 22-3](#).

Mode(s): Host

The USBRQPKTCOUNT[n] registers are shown in [Figure 22-55](#) and described in [Table 22-58](#).

Figure 22-55. USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[n])

15	13	12	0
Reserved		COUNT	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; - n = value after reset

Table 22-58. USB Request Packet Count in Block Transfer Endpoint n Registers (USBRQPKTCOUNT[n]) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	COUNT		Block Transfer Packet Count sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer. Note: This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

22.6.44 USB Receive Double Packet Buffer Disable Register (USBXDPKTBUFDIS), offset 0x340

The USB receive double packet buffer disable 16-bit register (USBTXIE) indicates which of the receive endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in [Section 22.3.1.1.1](#)).

Mode(s): Host Device

USBXDPKTBUFDIS is shown in [Figure 22-56](#) and described in [Table 22-59](#).

Figure 22-56. USB Receive Double Packet Buffer Disable Register (USBXDPKTBUFDIS)

15								4	3	2	1	0
Reserved									EP3	EP2	EP1	Rsvd
R-0									R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-59. USB Receive Double Packet Buffer Disable Register (USBXDPKTBUFDIS) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0 1	EP3 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
2	EP2	0 1	EP2 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
1	EP1	0 1	EP1 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
0	Reserved	0	Reserved

22.6.45 USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS), offset 0x342

The USB transmit double packet buffer disable 16-bit register (USBTXIE) indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see *Double-Packet Buffering* in [Section 22.3.1.1.1](#)).

Mode(s): Host Device

USBTXDPKTBUFDIS is shown in [Figure 22-57](#) and described in [Table 22-60](#).

Figure 22-57. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS)

15								4	3	2	1	0
Reserved									EP3	EP2	EP1	Rsvd
R-0									R/W-1	R/W-1	R/W-1	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-60. USB Transmit Double Packet Buffer Disable Register (USBTXDPKTBUFDIS) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Reserved
3	EP3	0 1	EP3 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
2	EP2	0 1	EP2 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
1	EP1	0 1	EP1 RX Double-Packet Buffer Disable Enables double-packet buffering. Disables double-packet buffering.
0	Reserved	0	Reserved

22.6.46 USB External Power Control Register (USBEPD), offset 0x400

The USB external power control 32-bit register (USBEPD) specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

Mode(s): Host Device

USBEPD is shown in [Figure 22-58](#) and described in [Table 22-61](#).

Figure 22-58. USB External Power Control Register (USBEPD)

31	Reserved															16
R-0																
15	Reserved										10	9	8			
R-0											PFLTACT					
R/W-0																
7	6	5	4	3	2	1	0									
Reserved	PFLTAEN	PFLTSEN	PFLTEN	Reserved	EPENDE	EPEN										
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0										

LEGEND: R/W = Read/Write; -n = value after reset

Table 22-61. USB External Power Control Register (USBEPD) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	PFLTACT	0h 1h 2h 3h	Power Fault Action. This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault. Unchanged. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. Tristate. USB0EPEN is undriven (tristate). Low. USB0EPEN is driven Low. High. USB0EPEN is driven High.
7	Reserved	0	Reserved
6	PFLTAEN	0 1	Power Fault Action Enable. This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal. Disabled. USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. Enabled. The USB0EPEN output is automatically changed to the state specified by the PFLTACT field.
5	PFLTSEN	0 1	Power Fault Sense. This bit specifies the logical sense of the USB0PFLT input signal that indicates an error condition. The complementary state is the inactive state. Low Fault. If USB0PFLT is driven Low, the power fault is signaled internally (if enabled by the PFLTEN bit). High Fault. If USB0PFLT is driven High, the power fault is signaled internally (if enabled by the PFLTEN bit).
4	PFLTEN	0 1	Power Fault Input Enable. This bit specifies whether the USB0PFLT input signal is used in internal logic. Not Used. The USB0PFLT signal is ignored. Used. The USB0PFLT signal is used internally
3	Reserved	0	Reserved

Table 22-61. USB External Power Control Register (USBEPD) Field Descriptions (continued)

Bit	Field	Value	Description
2	EPENDE		EPEN Drive Enable. This bit specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN field. When not driven, the EPEN field is ignored and the USB0EPEN signal is placed in a high-impedance state. The USB0EPEN signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers can bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.
		0	Not Driven. The USB0EPEN signal is high impedance.
		1	Driven. The USB0EPEN signal is driven to the logical value specified by the value of the EPEN field.
1-0	EPEN		External Power Supply Enable Configuration. This bit field specifies and controls the logical value driven on the USB0EPEN signal.
		0h	Power Enable Active Low. The USB0EPEN signal is driven Low if the EPENDE bit is set.
		1h	Power Enable Active High. The USB0EPEN signal is driven High if the EPENDE bit is set.
		2h	Power Enable High if VBUS Low. The USB0EPEN signal is driven High when the A device is not recognized.
		3h	Power Enable High if VBUS High. The USB0EPEN signal is driven High when the A device is recognized.

22.6.47 USB External Power Control Raw Interrupt Status Register (USBEPCRIS), offset 0x404

The USB external power control raw interrupt status 32-bit register (USBEPCRIS) specifies the unmasked interrupt status of the two-pin external power interface.

Mode(s): Host Device

USBEPCRIS is shown in [Figure 22-59](#) and described in [Table 22-62](#).

Figure 22-59. USB External Power Control Raw Interrupt Status Register (USBEPCRIS)

31	Reserved	1	0
			PF
	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-62. USB External Power Control Raw Interrupt Status Register (USBEPCRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	PF	0	USB Power Fault Interrupt Status. This bit is cleared by writing a 1 to the PF bit in the USBEPCISC register.
		0	A Power Fault status has been detected.
		1	An interrupt has not occurred.

22.6.48 USB External Power Control Interrupt Mask Register (USBEPICIM), offset 0x408

The USB external power control interrupt mask 32-bit register (USBEPICIM) specifies the interrupt mask of the two-pin external power interface.

Mode(s): Host Device

USBEPICIM is shown in [Figure 22-59](#) and described in [Table 22-62](#).

Figure 22-60. USB External Power Control Interrupt Mask Register (USBEPICIM)

31		1	0
Reserved			PF
R-0			R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-63. USB External Power Control Interrupt Mask Register (USBEPICIM) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	PF	0	USB Power Fault Interrupt Mask. The raw interrupt signal from a detected power fault is sent to the interrupt controller.
		1	A detected power fault does not affect the interrupt status.

22.6.49 USB External Power Control Interrupt Status and Clear Register (USBEPICISC), offset 0x40C

The USB external power control interrupt status and clear 32-bit register (USBEPICISC) specifies the unmasked interrupt status of the two-pin external power interface.

Mode(s): Host Device

USBEPICISC is shown in [Figure 22-61](#) and described in [Table 22-64](#).

Figure 22-61. USB External Power Control Interrupt Status and Clear Register (USBEPICISC)

31	Reserved	1	0
	R-0		PF
			R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-64. USB External Power Control Interrupt Status and Clear Register (USBEPICISC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF	0	USB Power Fault Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the USBEPICISC register. The PF bits in the USBEPCIRIS and USBEPCIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

22.6.50 USB Device RESUME Raw Interrupt Status Register (USBDRRIS), offset 0x410

The USB device RESUME raw interrupt status register (USBDRRIS) is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

Mode(s): Host Device

USBDRRIS is shown in [Figure 22-62](#) and described in [Table 22-65](#).

Figure 22-62. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)

31	Reserved	1	0
	R-0		RESUME
			R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-65. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF	0	RESUME Interrupt Status This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register. A RESUME status has been detected.
		1	An interrupt has not occurred.

22.6.51 USB Device RESUME Raw Interrupt Mask Register (USBDRIM), offset 0x414

The USB device RESUME raw interrupt status register (USBDRIM) is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

Mode(s): Host Device

USBDRIM is shown in [Figure 22-63](#) and described in [Table 22-66](#).

Figure 22-63. USB Device RESUME Raw Interrupt Status Register (USBDRRIS)

31	Reserved	1	0
	R-0		R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-66. USB Device RESUME Raw Interrupt Status Register (USBDRRIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	PF	0	RESUME Interrupt Mask
		0	The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the SUSPEND bit in the USBIS register is set).
		1	A detected RESUME does not affect the interrupt status.

22.6.52 USB Device RESUME Interrupt Status and Clear Register (USBDRISC), offset 0x418

The USB device RESUME interrupt status and clear register (USBDRRIS) is the raw interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

Mode(s): Host Device

USBDRISC is shown in [Figure 22-64](#) and described in [Table 22-67](#).

Figure 22-64. USB Device RESUME Interrupt Status and Clear Register (USBDRISC)

31	1	0
Reserved		RESUME
R-0		R/W1C

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-67. USB Device RESUME Interrupt Status and Clear Register (USBDRISC)
Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. Reset is 0x0000.000.
0	RESUME	0	RESUME Interrupt Status and Clear. This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the USBDRCRIS register.
		0	The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller.
		1	No interrupt has occurred or the interrupt is masked.

22.6.54 USB DMA Select Register (USBDMASEL), offset 0x450

The USB DMA select 32-bit register (USBDMASEL) specifies whether the unmasked interrupt status of the ID value is valid.

Mode(s):	Host	Device
----------	------	--------

USBDMASEL is shown in [Figure 22-66](#) and described in [Table 22-69](#).

Figure 22-66. USB DMA Select Register (USBDMASEL)

31				24	23		20	19		16
Reserved					DMACTX			DMACRX		
R/0					R/W-0			R/W-0		
15	12	11		8	7		4	3		0
DMABTX			DMABRX			DMAATX			DMAARX	
R/W-0			R/W-0			R/W-0			R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 22-69. USB DMA Select Register (USBDMASEL) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. Reset is 0x0000.000.
23-20	DMACTX		DMA C TX Select specifies the TX mapping of the third USB endpoint on DMA channel 5 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX
19-16	DMACRX		DMA C RX Select specifies the RX and TX mapping of the third USB endpoint on DMA channel 4 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX
15-12	DMABTX		DMA B TX Select specifies the TX mapping of the second USB endpoint on DMA channel 3 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX
11-8	DMABRX		DMA B RX Select Specifies the RX mapping of the second USB endpoint on DMA channel 2 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX
7-4	DMAATX		DMA A TX Select specifies the TX mapping of the first USB endpoint on DMA channel 1 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 TX
		2h	Endpoint 2 TX
		3h	Endpoint 3 TX

Table 22-69. USB DMA Select Register (USBDMASEL) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	DMAARX		DMA A RX Select specifies the RX mapping of the first USB endpoint on DMA channel 0 (primary assignment).
		0h	Reserved
		1h	Endpoint 1 RX
		2h	Endpoint 2 RX
		3h	Endpoint 3 RX

Universal Parallel Port (uPP)

This chapter discusses the features and functions of the Universal Parallel Port module (uPP).

Topic	Page
23.1 Introduction	2270
23.2 Configuring Device Pins	2271
23.3 Functional Description	2271
23.4 IO Interface and System Requirements	2274
23.5 Registers	2284

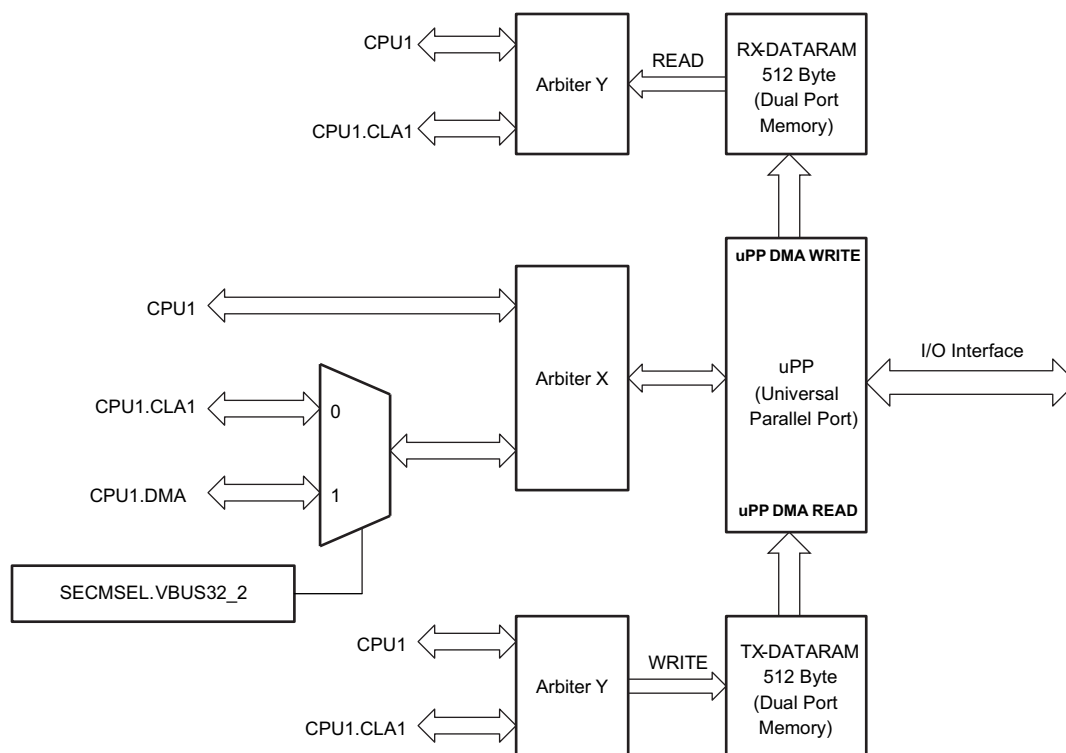
23.1 Introduction

The universal parallel port (uPP) peripheral is a high-speed parallel interface with dedicated data lines and minimal control signals. It is designed to interface cleanly with high-speed analog-to-digital converters (ADCs) or digital-to-analog converters (DACs) with 8-bit data width. It can also be interconnected with field-programmable gate arrays (FPGAs) or other uPP devices to achieve high-speed digital data transfer. It can operate in receive mode or transmit mode (simplex mode).

This peripheral includes an internal DMA controller to maximize throughput and minimize CPU overhead during high-speed data transmission. All uPP transactions use internal DMA to feed data to or retrieve data from the I/O channels. Even though there is only one I/O channel, the DMA controller includes two DMA channels to support data interleaved mode, in which all DMA resources service a single I/O channel.

On this device, uPP is a dedicated resource for CPU1 subsystem and CPU1, CPU1.CLA1 and CPU1.DMA have access to this module. There are two dedicated DATA RAMs (also known as MSG RAMs), each of 512B, tightly coupled with uPP module (one for each, TX and RX). These DATA RAMs are used to store bulk of data to avoid frequent interruption to CPU. Only CPU1 and CPU1.CLA1 has access to these DATA RAMs. Figure 23-1 shows the integration of uPP on this device.

Figure 23-1. uPP Integration



NOTE: In some other TI devices, uPP IP is also called the Radio Peripheral Interface (RPI) module.

23.1.1 Features Supported

- Supports mainstream high-speed data converters with parallel conversion interface.
- Supports mainstream high-speed streaming interface with frame START indication.
- Supports mainstream high-speed streaming interface with data ENABLE indication.
- Supports mainstream high-speed streaming interface with synchronization WAIT signal.
- Supports SDR (single-data-rate) or DDR (double-data-rate, interleaved) interface.
- Supports multiplexing of interleaved data in SDR transmit case.
- Supports de-multiplexing and multiplexing of interleaved data in DDR case.

- Supports I/O interface clock frequency up to 50 MHz for SDR case, and 25 MHz for DDR.
- Supports single-channel 8-bit input receive or output transmit mode.
- Max throughput is 50MB/s for pure read or pure write.
- Available as a DSP to FPGA general purpose streaming interface.

23.2 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.

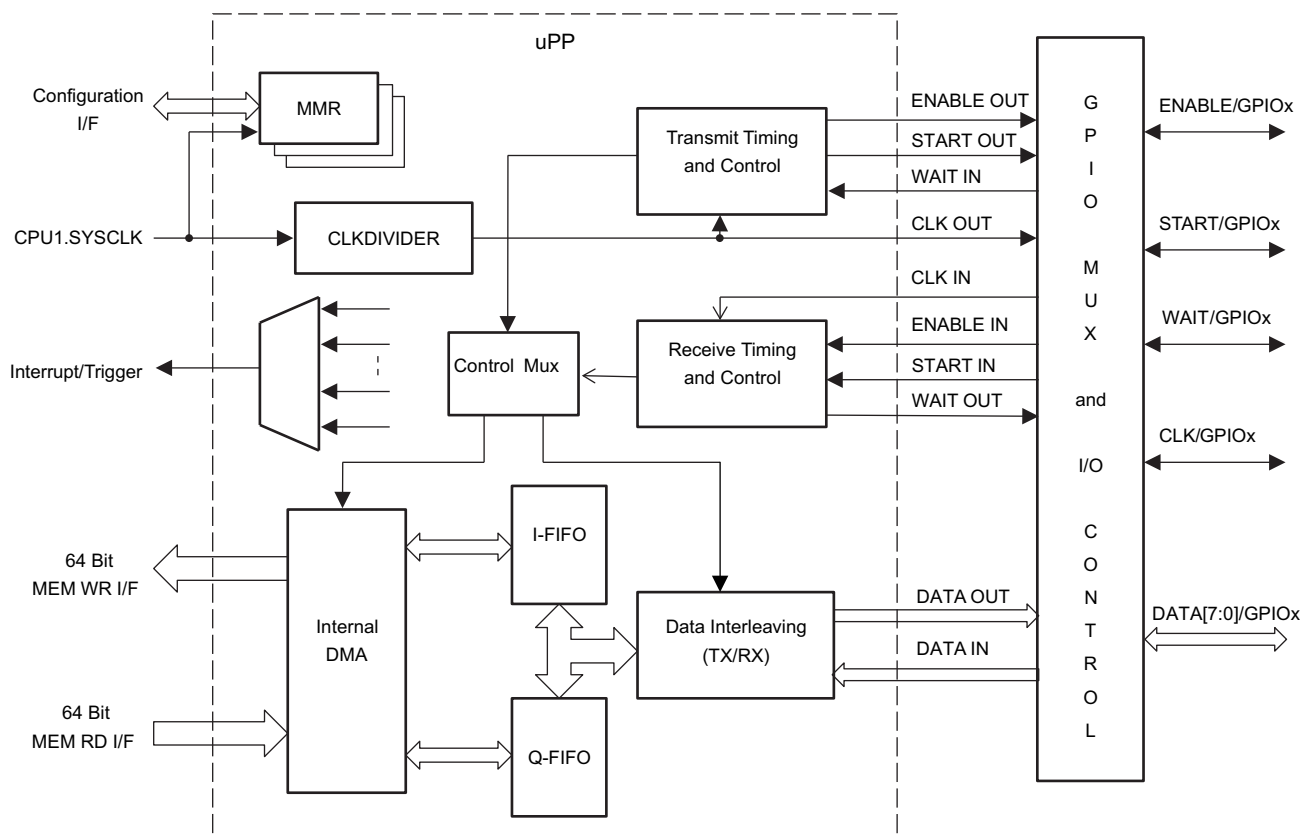
Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

23.3 Functional Description

23.3.1 Functional Block Diagram

Figure 23-2. Functional Block Diagram



The uPP IP supports internal FIFO to store the data to and from IOs. To alleviate the load of data transfer from system memory to and from the uPP FIFOs, the IP supports internal DMA. The internal DMA has two channels: channel I and channel Q.

23.3.2 Data Flow

The following naming conventions are followed regarding channel numbering:

- It has only one I/O channel and that has been labeled as “channel A”
- DMA channel are labeled as “channel I and channel Q”.

Figure 23-3 explains the data flow for receive in SDR mode and DDR non-demux mode. In this case, only one DMA channel (channel I) get used.

Figure 23-3. RX in SDR or DDR (non-demux) Mode

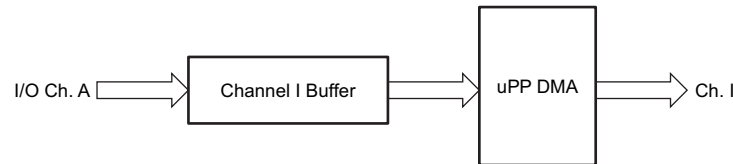


Figure 23-4 explains the data flow for receive in DDR demux mode. In this case, both DMA channels (channel I and channel Q) are used.

Figure 23-4. RX in DDR (demux) Mode

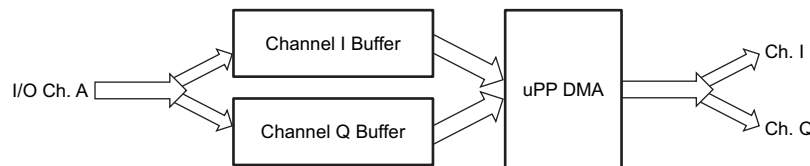


Figure 23-5 explains the data flow for transmit in SDR non-interleave or DDR non-demux mode. In this case, only one DMA channel (channel I) get used.

Figure 23-5. TX in SDR (non-interleave) or DDR (non-demux) Mode

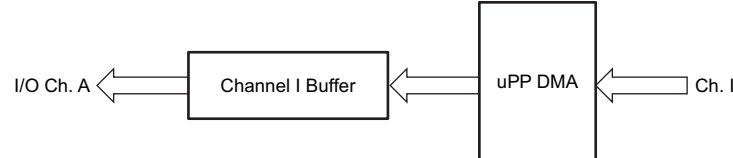
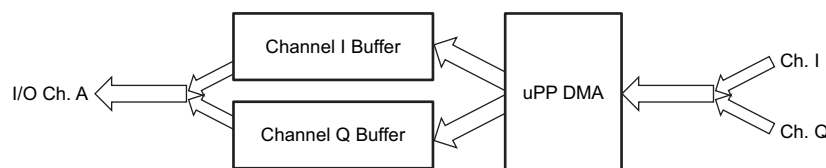


Figure 23-6 explains the data flow for transmit in SDR interleave or DDR demux mode. In this case, both DMA channels (channel I and channel Q) are used.

Figure 23-6. TX in SDR (interleave) or DDR (demux) Mode



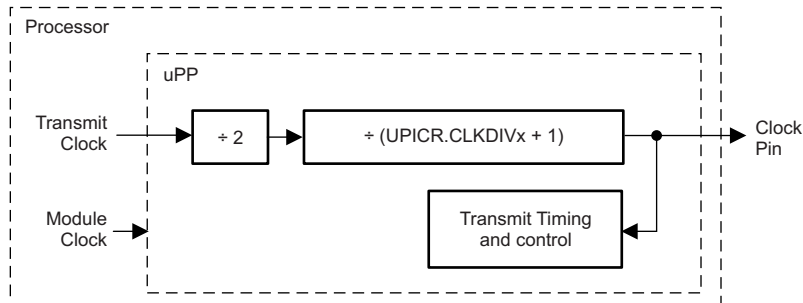
23.3.3 Clock Generation and Control

The uPP peripheral uses three different clocks:

- A module clock that controls its internal logic and CPU interface. This is driven by CPU1.SYSCCLK.
- A transmit clock (internally divided clock from the module clock) that runs the interface channel and drives the CLK pin out in transmit mode.
- A receive clock (input via CLK pin) that runs the interface channel in receive mode.

Figure 23-7 shows the output clock generation system in transmit mode.

Figure 23-7. IO Output Clock Generation for TX Mode

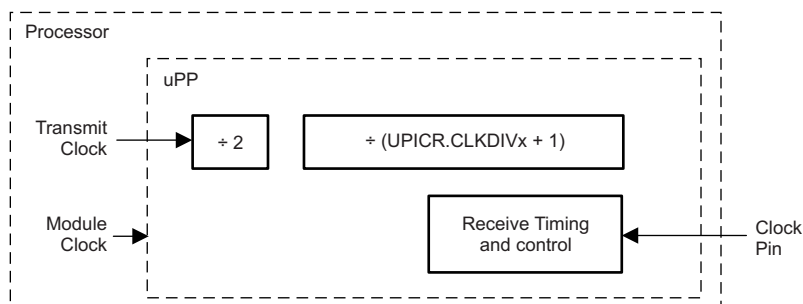


$$\text{Output Clock} = \text{Module Clock} / (2 \times (\text{CLKDIVn} + 1))$$

The fixed divisor restricts the maximum speed of the I/O clock to 1/2 the device CPU clock speed. For DDR mode operation, the CLK output frq must be one eighth (1/8) or less of module clock.

For receive mode, channel requires an external clock to drive its CLK pin. The incoming clock is not divided, and its maximum allowed speed is one fourth (1/4) the module clock speed for SDR mode and one eighth (1/8) the module clock speed for DDR mode. [Figure 23-8](#) shows the clock generation system for a channel configured in receive mode.

Figure 23-8. IO Input clock for RX Mode



23.4 IO Interface and System Requirements

The uPP module provides interfacing logic to external streaming data, both inbound and outbound flow of data are supported. The interface protocol is a simple streaming interface. It is build on top of the existing high-speed data converters, and captures most of the high-speed data converters protocol.

The input receive mode and output transmit mode protocol are symmetrical, and source synchronous, that is, clock is supplied by the transmitter and the same clock is used to receive the data.

Table 23-1 describes the functionality of all the input/output (IO) signals of uPP module.

Table 23-1. uPP Signal Description

Signal	Description
CLK	<ul style="list-style-type: none"> Transmit or receive clock Input in receive mode Output in transmit mode (Clock divider programmable)
START	<ul style="list-style-type: none"> Input in receive mode Output in transmit mode Polarity programmable Indicates first data word of each line (frame) This is optional signal and if this feature is not used, external receiver can ignore this signal.
ENABLE	<ul style="list-style-type: none"> Input when in receive mode Output in transmit mode Polarity programmable Indicates data received is valid, or data transmitted is valid Can be programmed to be ignored in receive mode but in transmit mode always drive active in sync with valid data.
WAIT	<ul style="list-style-type: none"> Input in transmit mode Output in receive mode but always drive inactive by uPP in this case Polarity programmable In transmit mode when active, indicates that target is not ready to receive data. uPP will stop transmitting data at next clock cycle right after Wait signal is sampled as active. uPP module can be programmed to ignored this in transmit mode. <p>NOTE: The WAIT needs to be asserted for one full cycle of IO clock by external device.</p>
DATA[7:0]	<ul style="list-style-type: none"> 8-bit inputs in receive mode 8-bit output in transmit mode

23.4.1 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. To determine how pin multiplexing affects the uPP peripheral, see the device-specific data manual.

23.4.2 Internal DMA Controller Description

The uPP peripheral includes an internal DMA controller separate from any device-level DMA. The internal DMA controller consists of two DMA channels, channel I and channel Q, which moves data to and from the uPP peripheral interface (I/O) channels in all operating modes. This section describes how to program the internal DMA channels.

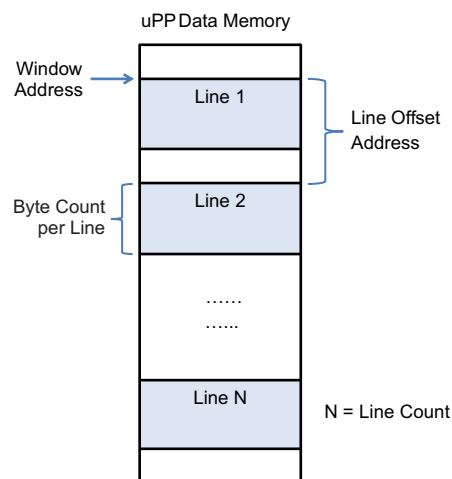
23.4.2.1 DMA Programming Concepts

The uPP internal DMA controller uses a simplified programming model similar to 2D transfers performed by any other DMA at system level. Each DMA channel may be configured with four parameters: window address, byte count, line count, and line offset address.

- **Window Address (CHxDESC0.ADDR)** - The location in uPP data memory of the first byte in the data buffer. When the uPP operates in receive mode, the DMA channel begins writing to this address as it takes incoming data from the uPP I/O channel. When the uPP operates in transmit mode, the DMA channel begins reading from this address and pass the data to the uPP I/O channel. The window address must be aligned to a 64-bit boundary (that is, the three LSBs must equal 0). Non-aligned addresses are automatically adjusted to a properly aligned value when written to configuration register.
- **Byte Count (CHxDESC1.BCNT)** - The number of bytes per line. The byte count must be an even number.
- **Line Count (UPxDESC1.LCNT)** - The number of lines per window. The total number of bytes transferred equals $B \times L$, where B is the byte count per line and L is the line count.
- **Line Offset Address (CHxDESC2.LOFFSET)** - The offset address between the first byte in successive lines. The line offset address cannot exceed 65528 (FFF8h) bytes, and must be aligned to a 64-bit boundary in memory (that is, the three LSBs must equal 0).

Figure 23-9 shows a typical DMA window defined by these parameters.

Figure 23-9. Structure of DMA Window and Lines in Memory



Certain values of the line offset address have special implications on the structure of the data buffer:

- **Line Offset Address = Byte Count** : Data buffer is a contiguous block in memory with size equal to (Line Count) x (Byte Count).
- **Line Offset Address = 0** : Data buffer consists of a single line, with total size equal to Byte Count. If the I/O channel is configured in transmit mode, this line is transmitted (Line Count) consecutive times before the DMA transfer completes. If the I/O channel is configured in receive mode, the buffer is repeatedly written and overwritten by incoming data.

To program a DMA transfer, write the appropriate fields in the DMA channel descriptor registers for DMA Channel I or for DMA Channel Q. If the associated I/O channel is initialized and idle, the DMA transfer and I/O transaction begins immediately. [Section 23.4.10](#) describes a step-by-step process for configuring the I/O channel and DMA channels to start a uPP transfer.

Each DMA channel allows a second descriptor to be queued while the previously programmed DMA transfer is still running. The PEND status bit in channel status register reports whether a new set of DMA parameters may be written to the DMA descriptor registers. Each DMA channel can have at most one active transfer and one queued transfer. This allows each I/O channel to perform uninterrupted, consecutive transactions across DMA transfer boundaries.

This DMA controller does not support automatically reloading DMA transfer descriptors. Each successive descriptor set must be explicitly written to the configuration registers by software. All uPP interrupt events originate in the internal DMA controller. [Section 23.4.6](#) lists and explains all uPP interrupt events.

The internal DMA controller always writes data in bursts of 64 bytes. However, DMA read operations have configurable burst size, which may be set per channel using the RDSIZEI and RDSIZEQ bits in the uPP threshold configuration register. A DMA channel waits until the specified number of bytes leaves its internal buffer before performing another burst read from memory.

Note that the TXSIZEA bit in threshold configuration register is not DMA parameters; instead, it control transmit thresholds for the uPP interface channel.

23.4.2.2 Data Interleave Mode

The data interleave mode is a special configuration that maps both DMA channels to a single interface I/O channel. There are two variants on data interleave mode, each with special conditions:

- Single Data Rate (SDR) Interleave
 - Single Data Rate (DRA = 0)
 - Transmit Only (MODE = 1)
 - SDR Transmit interleave enabled (SDRTXILA = 1)
- Double Data Rate (DDR) Interleave
 - Double Data Rate (DRA = 1)
 - Transmit or Receive (MODE = 0/1)
 - SDR Transmit interleave disabled (SDRTXILA = 0)
 - DDR interleave enabled (DEMUXA = 1)

In data interleave mode, I/O channel is associated with two data buffers, each serviced by its own DMA channel (I and Q). In SDR interleave mode, the START signal is used as a buffer selection line: START = 1 indicates that the current word comes from DMA Channel I; START = 0 indicates that the current word comes from DMA Channel Q. In DDR Interleave mode, the data buffers alternate every word beginning with Channel I: Channel I Word 0, Channel Q Word 0, Channel I Word 1, Channel Q Word 1, and so forth. [Section 23.4.10](#) shows signal diagrams for both data interleave modes.

23.4.3 Protocol Description

The uPP peripheral on this device has one I/O channel with 8 data lines, which can be configured to run in transmit or receive mode. A channel may also be configured to ignore certain control signals using the uPP interface configuration register. By default it uses all four control signals, unless otherwise configured.

23.4.3.1 DATA[7:0] Signals

DATA[7:0] comprise the channel's entire data bus. In transmit mode, these pins are outputs that transmit data supplied by the channel's associated DMA channel. While the channel is idle, their behavior depends on the TRISENA bit in IFCFG register. These pins can be configured to drive an idle value (TRISENA = 0, VALA field in the uPP interface idle value register (IFIVAL)) or be in a high-impedance state while idle (TRISENA = 1). In receive mode, these pins are inputs that provide data to the channel's associated DMA channel.

23.4.3.2 START Signal

The uPP transmitter asserts the START signal when it transfers the first word of a data line. A line is defined in terms of the channel's associated DMA channel; for more on DMA programming concepts, see [Section 23.4.2](#). The START signal is active-high by default, but its polarity is controlled by the STARTPOLA bit in IFCFG register.

In transmit mode, START is an output signal and is always driven, in receive mode, START is an input signal and may be disabled using the STARTA bit in IFCFG register. When the channel is configured in transmit mode with data interleave enabled (SDRTXILA = 1 in CHCTL register), the START signal function changes completely. The START signal now asserts on every data word that is provided by DMA Channel I. For this alternative behavior, see [Section 23.4.3.6](#).

23.4.3.3 ENABLE

The uPP transmitter asserts the ENABLE signal when it transfers a valid data word. The ENABLE signal is active-high by default, but its polarity is controlled by the ENAPOLA bit in IFCFG register. In transmit mode, ENABLE is an output signal and is always driven; in receive mode, ENABLE is an input signal and may be disabled using the ENAA bit in IFCFG register.

23.4.3.4 WAIT Signal

The WAIT signal allows the receiver to request a temporary halt in data transmission. When the receiver asserts WAIT, the transmitter responds by stopping transmission (starting with the next word) until WAIT is released. The receiver ignores all incoming data until WAIT is released. Once WAIT is released, the transmitter can resume transmission on the next word. [Section 23.4.3](#) shows WAIT signal timing. The WAIT signal is active-high by default, but its polarity is controlled by the WAITPOLA bit in IFCFG register. In transmit mode, WAIT is an input signal and may be disabled using the WAITA bit in IFCFG register; in receive mode, WAIT is an output signal and always driven inactive by uPP.

23.4.3.5 CLOCK Signal

The uPP transmitter drives the CLOCK signal to align all other uPP signals. By default, other signals align on the rising edge of CLOCK, but its polarity is controlled by the CLKINVA bit in IFCFG register. The active edge(s) of CLOCK should always slightly precede transitions of other uPP signals. In transmit mode, CLOCK is an output signal; in receive mode, CLOCK is an input signal. For more information on clock generation and allowed frequencies, see [Section 23.4.2](#).

23.4.3.6 Signal Timing Diagrams

In the following diagrams, signals are marked (I) to indicate that they are inputs to the uPP peripheral and (o) to indicate that they are outputs from the uPP peripheral. Data words from a single DMA channel are designated Dx, while data words that must come from a specific DMA channel are designated Ix or Qx to indicate DMA Channel I or Q, respectively. For more information on DMA channels and data interleave mode, see [Section 23.4.2](#).

All signal diagrams are drawn with signal polarities in their default states. All signals except DATA are independently configurable in the uPP interface configuration register (IFCFG).

Figure 23-10. uPP Receive in SDR Mode

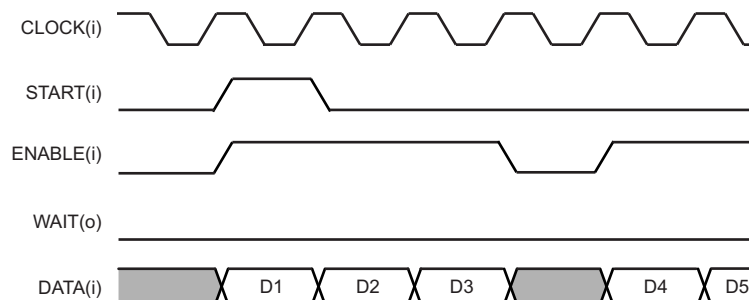
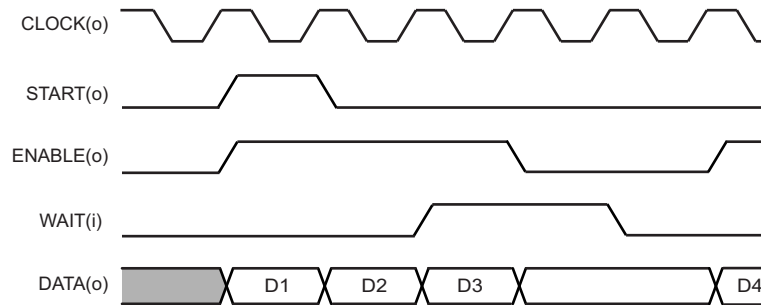
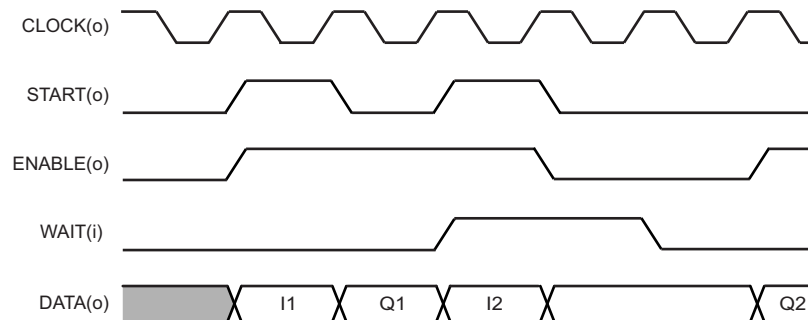
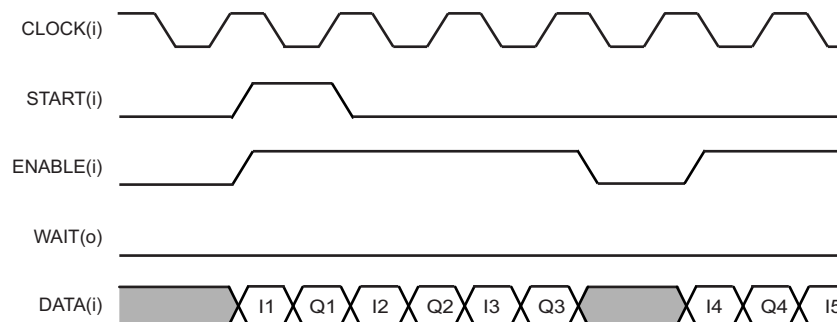
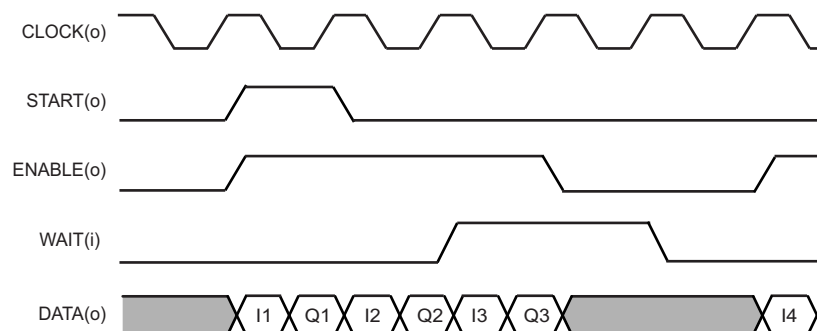


Figure 23-11. uPP Transmit in SDR Mode

Figure 23-12. uPP Transmit in SDR Mode – Interleaving

Figure 23-13. uPP Receive DDR Case

Figure 23-14. uPP Transmit DDR Case


NOTE: START asserts on every data word from DMA Channel I in interleaving mode.

23.4.4 Data Format

On this device, uPP only support 8-bit interface hence no data packing is performed. User in this case, will have to make appropriate connection externally, if needed. For example, if a 4-bit ADC is used, then the user would have to tie the upper 4-bit of the uPP data inputs to '0' or '1' as needed. If external FPGA is connected, then the user would have to make sure that 8-bit data (or appropriate tie-offs) is driven out from the FPGA. By default uPP supports little endian data format but to support the data format of one of the TI device (CC1260), a configuration bit is provided to select 16bit big endian mode also.

Figure 23-15 and Figure 23-16 describe how data goes out and comes in based on these modes.

Figure 23-15. uPP Tx Data Pattern in Non-Interleaved Mode

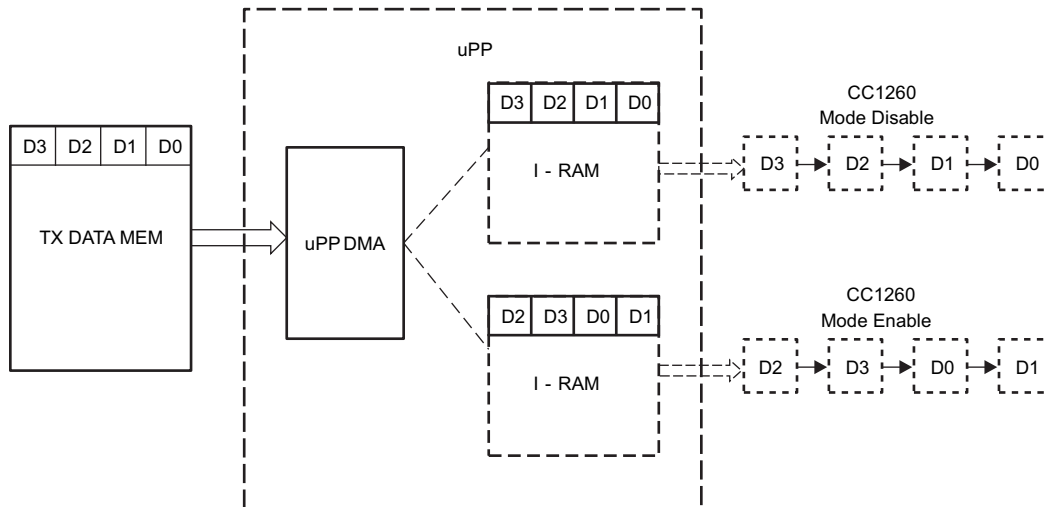
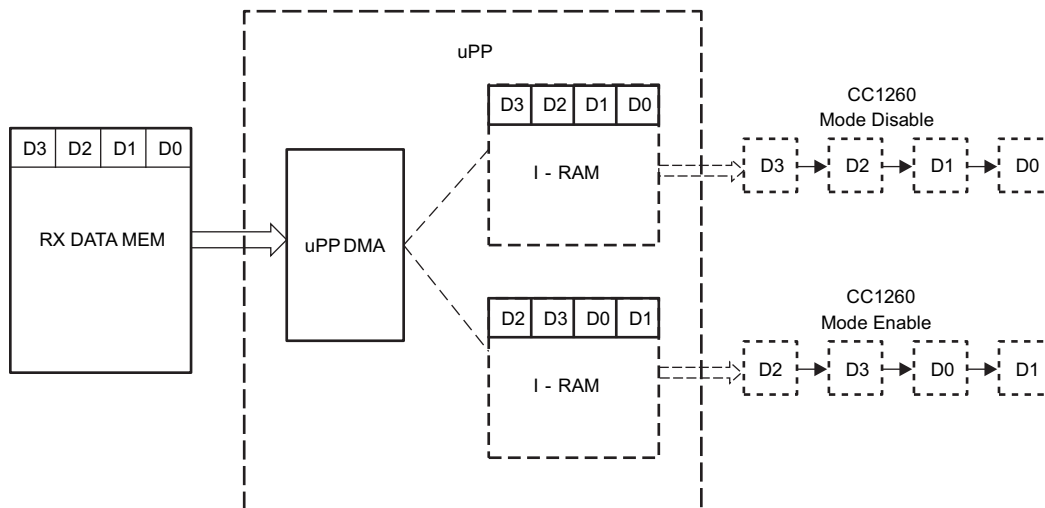


Figure 23-16. uPP Rx Data Pattern in Non-Interleaved Mode



23.4.5 Reset Considerations

23.4.5.1 Software Reset

Software reset clears the uPP internal state machines but does not reset the contents of the uPP registers. The following procedure performs a software reset on the uPP peripheral:

1. Write the PEREN bit in the uPP PERCTL register to 0 (disables the uPP).
2. Poll the DMAST bit in uPP PERCTL register for activity; wait until DMA controller is inactive and idle.
3. Write the SOFTRST bit in uPP PERCTL register to 1 (places uPP in software reset).

4. Write the SOFTRST bit in uPP PERCTL register to 0 to (brings uPP out of software reset).

NOTE: In receive mode, uPP IO input clock must be running to have effect of soft reset.

23.4.5.2 Hardware Reset

When the processor XRSn (or CPU1.SYSRSn) pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the uPP state machines are reset, and the uPP registers are forced to their default states. For default states, see the register section.

NOTE: There is no SW method to apply a hard reset to the UPP module on this device.

23.4.6 Interrupt Support

The uPP peripheral generates multiple interrupt events, all tied to internal DMA Channels I and Q. The uPP peripheral automatically combines all interrupt events into a single chip-level interrupt driven to CPU1 and CPU1.CLA1. Individual events may be enabled using the uPP interrupt enable set register (INTENSET) and disabled using the uPP interrupt enable clear register (INTENCLR). Only enabled events generate interrupts and assert bits in the enabled interrupt status register (ENINTST). Disabled events do not generate interrupts but do assert bits in the raw interrupt status register (RAWINTST). An interrupt service routine (ISR) may be assigned to handle uPP CPU-level interrupts using the interrupt controller module. If uPP events occur in close proximity to one another, a single CPU interrupt (and a single call to the ISR) may represent multiple interrupt events. Thus, the uPP ISR must meet certain structural requirements:

- The ISR must be able to handle multiple events before returning
- The ISR must handle any subsequent events that occur after it is called but before it returns

Like CPU ISR, CLA tasks can be assigned based on these interrupt events.

23.4.6.1 End of Line (EOL) Event

This event occurs each time when DMA channel reaches the end of a line in the data window. Note that if the associated uPP interface channel is operating in transmit mode, this event may occur before the line's final bytes are actually transmitted over the data pins.

For small line size and fast data transfer, it is possible to "miss" EOL events if they occur faster than the user's code can handle them. This does not hinder uPP operation; the uPP peripheral continue processing data uninterrupted until the EOW event or some error condition is encountered.

23.4.6.2 End of Window (EOW) Event

This event occurs when the DMA channel reaches the end of its current data window. Note that if the associated uPP interface channel is operating in transmit mode, this event may occur shortly before the window's final bytes are actually transmitted over the data pins.

When an EOW event occurs, the DMA channel automatically begins the next DMA transfer if one has been pre-programmed into the channel descriptor registers. If no new transfer is preprogrammed, the DMA channel becomes idle. For small window size and fast data transfer, code overhead may make it impossible to maintain a constant flow of data through the uPP interface channel. This problem can be solved by increasing the DMA window size or decreasing the peripheral clock speed.

23.4.6.3 Underrun or Overflow (UOR) Event

This event occurs when the DMA channel fails to keep up with incoming or outgoing data on its associated interface channel. Typically, this error indicates that background system activity has interfered with normal operation of the peripheral. It does not occur simply when a channel is allowed to idle. After encountering this error, the uPP peripheral should be reset when this event occurs.

This error should primarily occur when operating the uPP at high speed with significant system loading. To avoid this error, run the uPP at slower speeds or reduce background activity, such as non-uPP peripheral or DMA transactions. Additional tuning tips are given in [Section 23.4.10.1](#).

23.4.6.4 DMA Programming Error (DPE) Event

This event occurs when the DMA channel descriptors are programmed while its PEND bit in the uPP DMA channel status register is set to 1. A channel's descriptors should only be programmed while the channel's PEND bit is cleared to 0.

23.4.7 Emulation Considerations

The uPP peripheral stops running if any of three conditions are met:

- Peripheral Disable - EN bit in the uPP peripheral control register (PERCTL) is 0.
- Clock Stop – Clock to uPP is disabled using PCLKCR.
- Emulation Suspend - JTAG emulator halts chip while FREE = 0 and SOFT = 1 in uPP PERCTL register.

For other settings of FREE and SOFT, the uPP peripheral continues running during emulation halt. When the uPP encounters a stop condition, it completes the current DMA burst transaction (if one is active) before stopping.

I/O channel, configured in transmit mode, immediately places its pins in a high-impedance state and preserves the state of its internal state machines. Unless some reset event occurs (see [Section 23.4.5](#)), the channel can resume where it left off when the stop condition is cleared. I/O channel configured in receive mode only captures one additional data word. Further incoming data words are ignored as long as the stop condition persists.

23.4.8 Transmit and Receive FIFOs

Each of the uPP peripheral I/O channels has a 512-byte FIFO. In receive mode, the FIFO is divided into eight 64-byte blocks. In transmit mode, the FIFO is divided into blocks that can be set to 64, 128, or 256 bytes, configured by the TXSIZEA field in the uPP threshold configuration register (THCFG).

Transmission will not begin until the channel has loaded enough data to fill at least one full FIFO block. The internal DMA channels may also be configured to use a read threshold of 64, 128, or 256 bytes using the RDSIZEI or RDSIZEQ field in uPP THCFG register. The DMA write threshold is fixed at 64 bytes.

23.4.9 Transmit and Receive Data (MSG) RAM

On this device, uPP internal DMA doesn't have access to system memories. Instead, there are two dedicated DATA RAMs (also called as MSGRAM) that have been provided: one for each RX and TX; each of these DATA RAMs are 512B. Since C28 CPU/CLA operates in 16-bit addressing mode, whereas, uPP internal DMA operates in byte addressing mode, the addresses for these RAMs are different for different views.

[Table 23-2](#) describes the addresses for these RAMs for two different views.

Table 23-2. CPU/CLA/uPP-DMA Address Map

Data (MSG)RAM	CPU/CLA Address		uPP DMA (programming) Address	
	Start Address	End Address	Start Address	End Address
TX DATA RAM	0x6C00	0x6CFF	0x6C00	0x6DFF
RX DATA RAM	0x6E00	0x6EFF	0x7000	0x71FF

[Table 23-3](#) describes the different access type for both the DATA(MSG) RAMs.

Table 23-3. CPU/CLA/uPP-DMA Address Map

Master/Data RAM	TX DATA(MSG) RAM		RX DATA(MSG) RAM	
	Read	Write	Read	Write
CPU1	Yes	Yes	Yes	Yes (Debug Mode Only)
CPU1.CLA1	Yes	Yes	Yes	No
uPP-DMA	Yes	No	No	Yes

NOTE: The message RAMs clocks are gated off when uPP clock is gated by the PCLKCR configuration. Thus, the message RAMs are not accessible by CPU/CLA when RPI clock is disabled by CPU.

23.4.10 Initialization and Operation

This section provides step-by-step instructions for initializing and running the uPP peripheral in various modes. These instructions are given assuming that the device has just come out of a power-on reset (POR) state.

1. Apply the appropriate pin multiplexing settings. For more information, see the device-specific data manual, and (or) pin multiplexing utility.
2. Enable the clocks to the uPP peripheral. For more information, see the Clock Control section of this TRM.
3. Wait for few (~32) device clock cycles, and then clear the “Soft Reset” bit to 0 to bring the module out of reset.
4. Program the following uPP configuration registers: CHCTL, IFCFG, IFIVAL, THCFG.
 - (a) uPP Channel Control Register (data format, SDR/DDR, TX/RX, single/dual channel, interleave, demux)
 - (b) uPP Interface configuration Register (IO signal enable, polarity, clock divisor)
 - (c) uPP Interface idle value register (to drive value in idle mode for Tx)
 - (d) uPP Threshold configuration register (TX Size, DMA read burst size)
5. Program the uPP interrupt enable set register to interrupt generation for the desired events. Register an interrupt service routine (ISR) if desired; otherwise, polling is required.
6. Set the “PerEn” bit in the uPP peripheral control register (PCR) to 1 to turn on the uPP peripheral.
7. Allocate or initialize data buffers for use with uPP.
8. Program the DMA channels with their first transfers using the uPP DMA channel descriptor register0, 1 and 2 (byte count, line count, line offset).
9. Once the DMA descriptors are written, the module will start to receive and transmit data, and perform DMA transfers. Below is a simple description of how the DMA works.
10. Watch for interrupt events. Reprogram the DMA as necessary. (check that the PEND bit in the uPP DMA channel status is ‘0’).
 - (a) If polling, check uPP Interrupt enabled status register. Reading a bit as 1 indicates the corresponding event has occurred. Write the corresponding bit with 1 to clear.
 - (b) If using ISR, check ENINTST inside your ISR

Once the channel is started, the initial latency between the time when the first data is received or transmitted on the pin, and the time when the first DMA burst is transferred on the CBA bus, is unpredictable. This initial latency is determined by several factors:

- Clock ratio between system clock and uPP clock
- Various threshold values being programmed
- Chip-level traffic activities level

However, once the data is moving, all subsequent data movement is continuous (streaming). If such data movement cannot be maintained, then DMA under-run or over-flow situation may occur.

23.4.10.1 System Tuning Tips

The uPP peripheral can operate at high speed and transfer data at a very high rate. When operating the uPP near its upper limits, tuning certain parameters can help decrease the incidence of errors and the software overhead incurred servicing uPP data. [Table 23-4](#) lists several parameters that can be useful in system tuning. A parameter is defined as a “coarse” adjustment, if changing the parameter directly alters the peripheral throughput. A “fine” adjustment does not change the peripheral throughput, but it does affect general system performance.

Table 23-4. uPP Parameters Useful for System Tuning

Parameter	Register	Register Field	Edge Value	Safe Value	Description
Data Rate	CHCTL	DRA	1	0	Double data rate increases data transfer by a factor of 2 and greatly increases system loading for the same clock divisor. This is a coarse adjustment and is probably fixed due to design constraints.
Clock Division	IFCFG	CLKDIVA	0	1+	Increasing clock division is the most straight-forward way to decrease system loading. This is a coarse adjustment; the difference between CLKDIVx = 0 and 1 is the same (in terms of data rate) as the difference between single and double data rate.
DMA Read Burst Size	THCFG	RDSIZEQ RDSIZEI	0	3h	Increasing the DMA read threshold decreases system loading by generating fewer, larger DMA events. This is a fine adjustment.
DMA Line Size, Count	CHxDESC 1	LCNT BCNT	0		⁽¹⁾ Condensing uPP transfers into fewer, larger lines generates fewer end-of-line interrupts and, thus, invokes fewer ISR calls. This is a fine adjustment.
Total Transfer Size	CHxDESC 1	LCNT BCNT	⁽¹⁾	⁽¹⁾	Performing many small uPP transfers can require excessive software overhead (programming DMA descriptors, handling interrupts, and so forth) at high data rates. This is a fine adjustment.

⁽¹⁾ (1) These values vary per application. One example could be a 16-KB transfer. The same total data could be transferred as 16 1-KB lines or 2 8-KB lines.

23.5 Registers

23.5.1 UPP Base Addresses

Table 23-5. UPP Base Address Table

Device Registers	Register Name	Start Address	End Address
UppRegs ⁽¹⁾	UPP_REGS	0x0000_6200	0x0000_62FF

⁽¹⁾ Only available on CPU1.

23.5.2 UPP_REGS Registers

Table 23-6 lists the memory-mapped registers for the UPP_REGS. 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. UPP_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	PID	Peripheral ID Register		Go
2h	PERCTL	Peripheral Control Register		Go
8h	CHCTL	General Control Register		Go
Ah	IFCFG	Interface Configuration Register		Go
Ch	IFIVAL	Interface Idle Value Register		Go
Eh	THCFG	Threshold Configuration Register		Go
10h	RAWINTST	Raw Interrupt Status Register		Go
12h	ENINTST	Enable Interrupt Status Register		Go
14h	INTENSET	Interrupt Enable Set Register		Go
16h	INTENCLR	Interrupt Enable Clear Register		Go
20h	CHIDESC0	DMA Channel I Descriptor 0 Register		Go
22h	CHIDESC1	DMA Channel I Descriptor 1 Register		Go
24h	CHIDESC2	DMA Channel I Descriptor 2 Register		Go
28h	CHIST0	DMA Channel I Status 0 Register		Go
2Ah	CHIST1	DMA Channel I Status 1 Register		Go
2Ch	CHIST2	DMA Channel I Status 2 Register		Go
30h	CHQDESC0	DMA Channel Q Descriptor 0 Register		Go
32h	CHQDESC1	DMA Channel Q Descriptor 1 Register		Go
34h	CHQDESC2	DMA Channel Q Descriptor 2 Register		Go
38h	CHQST0	DMA Channel Q Status 0 Register		Go
3Ah	CHQST1	DMA Channel Q Status 1 Register		Go
3Ch	CHQST2	DMA Channel Q Status 2 Register		Go
40h	GINTEN	Global Peripheral Interrupt Enable Register		Go
42h	GINTFLG	Global Peripheral Interrupt Flag Register		Go
44h	GINTCLR	Global Peripheral Interrupt Clear Register		Go
46h	DLYCTL	IO clock data skew control Register		Go

23.5.2.1 PID Register (Offset = 0h) [reset = 44231100h]

PID is shown in [Figure 23-17](#) and described in [Table 23-7](#).

Peripheral ID Register

Figure 23-17. PID Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVID																															
R-44231100h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-7. PID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REVID	R	44231100h	Module revision id.

23.5.2.2 PERCTL Register (Offset = 2h) [reset = 0h]

PERCTL is shown in [Figure 23-18](#) and described in [Table 23-8](#).

Peripheral Control Register

Figure 23-18. PERCTL 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
DMAST	RESERVED		SOFTIRST	PEREN	RTEMU	SOFT	FREE
R-0h	R=0-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-8. PERCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R=0	0h	Reserved
7	DMAST	R	0h	DMA state machine status. 0: Idle 1: DMA Burst transaction is active.
6-5	RESERVED	R=0	0h	Reserved
4	SOFTIRST	R/W	0h	This bit reset all the state machines and certain memory elements inside the RPI module immediately. Software can write this bit to '1', and later write '0' to bring the RPI module out of reset. Note that MMR are NOT reset, except for Interrupt-Raw-Status Register. This reset can be used to recover the RPI from an error condition. To make sure that a graceful or fail-safe reset is performed, software can first disable the "PerEn" bit, then poll the DMAStatus bit to make sure that all pending VBUSP DMA burst are completed, then perform a software reset. 0: De-assert the reset (out of reset). 1: Assert the reset (in to reset).
3	PEREN	R/W	0h	This bit can be used to disable or suspend the RPI module. When this bit is programmed to be disabled, the RPI will be stopped (suspended) after all current DMA activity are completed. 0: Disable/Suspend the uPP module. 1: Enable/Resume the uPP module.
2	RTEMU	R/W	0h	0: Real Time emulation is disable. Module gets suspended when CPU is suspended. 1: Real Time emulation is enable.
1	SOFT	R/W	0h	0: Hard Stop. 1: Soft Stop.
0	FREE	R/W	0h	0: Software controlled. 1: Free Running.

23.5.2.3 CHCTL Register (Offset = 8h) [reset = 0h]

CHCTL is shown in [Figure 23-19](#) and described in [Table 23-9](#).

General Control Register

Figure 23-19. CHCTL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-9. CHCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R=0	0h	Reserved
30-17	RESERVED	R/W	0h	Reserved
16	DRA	R/W	0h	Data rate control. 0: Single Data Rate (SDR). 1: Double Data Rate (DDR).
15-5	RESERVED	R/W	0h	Reserved
4	DEMUXA	R/W	0h	DDR de-multiplexing mode (This bit is only valid for DDR mode): 0: De-multiplexing is disable. 1: De-multiplexing is Enable (split data into 2 DMA channels).
3	SDRTXILA	R/W	0h	Tx SDR interleave mode (This bit is only valid for SDR mode): 0: Interleaving is disable. 1: Interleaving is Enable (split data into 2 DMA channels).
2	RESERVED	R/W	0h	Reserved
1-0	MODE	R/W	0h	Operating mode: 00: Pure input receive mode. 01: Pure output transmit mode. 10: Reserved. 11: Reserved.

23.5.2.4 IFCFG Register (Offset = Ah) [reset = 0h]

IFCFG is shown in [Figure 23-20](#) and described in [Table 23-10](#).

Interface Configuration Register

Figure 23-20. IFCFG Register

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R=0-0h		R/W-0h					
23	22	21	20	19	18	17	16
RESERVED		RESERVED					
R=0-0h		R/W-0h					
15	14	13	12	11	10	9	8
RESERVED		TRISENA	CLKINVA	CLKDIVA			
R=0-0h		R/W-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED		WAITA	ENAA	STARTA	WAITPOLA	ENAPOLA	STARTPOLA
R=0-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-10. IFCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R=0	0h	Reserved
29-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R=0	0h	Reserved
21-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R=0	0h	Reserved
13	TRISENA	R/W	0h	Transmit mode output tri-state control: 0: Disable. RPI will not tri-state during idle time and will drive values from the Interface Idle Value Register. 1: Enable. RPI will tri-state during idle time. Idle time is the time before and between Window transfers.
12	CLKINVA	R/W	0h	Clock inversion: 0: Clock is not inverted 1: Clock is inverted When in Rx mode, the clock will be treated as inverted if enabled. When in Tx mode, the clock will be inverted before going out of the pin.
11-8	CLKDIVA	R/W	0h	Clock divider for transmit mode: $TX_IOCLK = CHIP_CLK / 2(N+1)$ Where 'N' is the value programmed into this field.
7-6	RESERVED	R=0	0h	Reserved
5	WAITA	R/W	0h	Enable Usage of WAIT signal: 0: Disable (Tx: ignore wait) 1: Enable (Tx: honor wait) This bit is only valid for transmit mode, receive mode always drive WAIT signal inactive (except for stop_run situation).

Table 23-10. IFCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	ENAA	R/W	0h	Enable Usage of ENABLE (WRITE) signal: 0: Disable (Rx: ignore enable) 1: Enable (Rx: honor enable) This bit is only valid for receive mode, transmit mode always drive ENABLE signal active.
3	STARTA	R/W	0h	Enable Usage of START (SELECT) signal: 0: Disable (Rx: ignore start) 1: Enable (Rx: honor start) This bit is only valid for receive mode, transmit mode always drive START signal active.
2	WAITPOLA	R/W	0h	Polarity of WAIT signal: 0: Active High. 1: Active Low.
1	ENAPOLA	R/W	0h	Polarity of ENABLE(WRITE) signal: 0: Active High. 1: Active Low.
0	STARTPOLA	R/W	0h	Polarity of START(SELECT) signal: 0: Active High. 1: Active Low.

23.5.2.5 IFIVAL Register (Offset = Ch) [reset = 0h]

IFIVAL is shown in [Figure 23-21](#) and described in [Table 23-11](#).

Interface Idle Value Register

Figure 23-21. IFIVAL Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-11. IFIVAL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	0h	Reserved
15-9	RESERVED	R/W	0h	Reserved
8-0	VALA	R/W	0h	When in transmit mode, this field holds the value that will be driven out when the channel is idle. This includes both situations when TRISEN field of the RPI Interface Configuration Register is enabled, and when wait-state is inserted by the external receiver (WAIT asserted).

23.5.2.6 THCFG Register (Offset = Eh) [reset = 0h]

THCFG is shown in [Figure 23-22](#) and described in [Table 23-12](#).

Threshold Configuration Register

Figure 23-22. THCFG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-12. THCFG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R=0	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-18	RESERVED	R=0	0h	Reserved
17-16	TXSIZEA	R/W	0h	I/O Transmit Threshold: 00: 64 Byte 01: 128 Byte 10: Reserved 11: 256 Byte The uPP will hold off transmitting until this threshold is reach in the transmit buffer. The following programming limitation applies: If TX_SIZE = 64B, there is no programming limitation. If TX_SIZE = 128B, descriptor Byte-Count must > 64B. If TX_SIZE = 256B, descriptor Byte-Count must > 192B
15-10	RESERVED	R=0	0h	Reserved
9-8	RDSIZEQ	R/W	0h	DMA Read Threshold for DMA Channel Q: 00: 64 Byte 01: 128 Byte 10: Reserved 11: 256 Byte DMA read burst is based on this value (same as FIFO block size). Note: DMA Write Threshold (write burst) is fixed at 64B.
7-2	RESERVED	R=0	0h	Reserved

Table 23-12. THCFG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	RDSIZEI	R/W	0h	DMA Read Threshold for DMA Channel I: 00: 64 Byte 01: 128 Byte 10: Reserved 11: 256 Byte DMA read burst is based on this value (same as FIFO block size). Note: DMA Write Threshold (write burst) is fixed at 64B.

23.5.2.7 RAWINTST Register (Offset = 10h) [reset = 0h]

RAWINTST is shown in [Figure 23-23](#) and described in [Table 23-13](#).

Raw Interrupt Status Register

Figure 23-23. RAWINTST 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			EOLQ	EOWQ	RESERVED	UOEQ	DPEQ
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED			EOLI	EOWI	RESERVED	UOEI	DPEI
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-13. RAWINTST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R=0	0h	Reserved
12	EOLQ	R/W	0h	Interrupt raw status for end-of-line condition: 0: No event. 1: End Of Line event happened.
11	EOWQ	R/W	0h	Interrupt raw status for end-of-line condition: 0: No event. 1: End Of Window event happened.
10	RESERVED	R/W	0h	Reserved
9	UOEQ	R/W	0h	Interrupt raw status for DMA under-run or over-run : 0: No event. 1: Under-run/Over-run event happened. Over-run in receiving or Under-Run in transmitting.
8	DPEQ	R/W	0h	Interrupt raw status for DMA programming error: 0: No event. 1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.
7-5	RESERVED	R=0	0h	Reserved
4	EOLI	R/W	0h	Interrupt raw status for end-of-line condition: 0: No event. 1: End Of Line event happened.
3	EOWI	R/W	0h	Interrupt raw status for end-of-line condition: 0: No event. 1: End Of Window event happened.
2	RESERVED	R/W	0h	Reserved
1	UOEI	R/W	0h	Interrupt raw status for DMA under-run or over-run : 0: No event. 1: Under-run/Over-run event happened. Over-run in receiving or Under-Run in transmitting.

Table 23-13. RAWINTST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DPEI	R/W	0h	Interrupt raw status for DMA programming error: 0: No event. 1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.

23.5.2.8 ENINTST Register (Offset = 12h) [reset = 0h]

ENINTST is shown in [Figure 23-24](#) and described in [Table 23-14](#).

Enable Interrupt Status Register

Figure 23-24. ENINTST 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			EOLQ	EOWQ	RESERVED	UOEQ	DPEQ
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED			EOLI	EOWI	RESERVED	UOEI	DPEI
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-14. ENINTST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R=0	0h	Reserved
12	EOLQ	R/W	0h	Interrupt enable status for end-of-line condition: 0: No event. 1: End Of Line event happened.
11	EOWQ	R/W	0h	Interrupt enable status for end-of-line condition: 0: No event. 1: End Of Window event happened.
10	RESERVED	R/W	0h	Reserved
9	UOEQ	R/W	0h	Interrupt enable status for DMA under-run or over-run : 0: No event. 1: Under-run/Over-run event happened. Over-run in receiving or Under-Run in transmitting.
8	DPEQ	R/W	0h	Interrupt enable status for DMA programming error: 0: No event. 1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.
7-5	RESERVED	R=0	0h	Reserved
4	EOLI	R/W	0h	Interrupt enable status for end-of-line condition: 0: No event. 1: End Of Line event happened.
3	EOWI	R/W	0h	Interrupt enable status for end-of-line condition: 0: No event. 1: End Of Window event happened.
2	RESERVED	R/W	0h	Reserved
1	UOEI	R/W	0h	Interrupt enable status for DMA under-run or over-run : 0: No event. 1: Under-run/Over-run event happened. Over-run in receiving or Under-Run in transmitting.

Table 23-14. ENINTST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DPEI	R/W	0h	Interrupt enable status for DMA programming error: 0: No event. 1: DMA programming error (Writing of DMA descriptors while PENDING bit is active) occurred.

23.5.2.9 INTENSET Register (Offset = 14h) [reset = 0h]

INTENSET is shown in [Figure 23-25](#) and described in [Table 23-15](#).

Interrupt Enable Set Register

Figure 23-25. INTENSET 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			EOLQ	EOWQ	RESERVED	UOEQ	DPEQ
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED			EOLI	EOWI	RESERVED	UOEI	DPEI
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-15. INTENSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R=0	0h	Reserved
12	EOLQ	R/W	0h	Interrupt enable for end-of-line condition: 0: Interrupt is disable. 1: Interrupt is enable.
11	EOWQ	R/W	0h	Interrupt enable for end-of-line condition: 0: Interrupt is disable. 1: Interrupt is enable.
10	RESERVED	R/W	0h	Reserved
9	UOEQ	R/W	0h	Interrupt enable for DMA under-run or over-run : 0: Interrupt is disable. 1: Interrupt is enable.
8	DPEQ	R/W	0h	Interrupt enable for DMA programming error: 0: Interrupt is disable. 1: Interrupt is enable.
7-5	RESERVED	R=0	0h	Reserved
4	EOLI	R/W	0h	Interrupt enable for end-of-line condition: 0: Interrupt is disable. 1: Interrupt is enable.
3	EOWI	R/W	0h	Interrupt enable for end-of-line condition: 0: Interrupt is disable. 1: Interrupt is enable.
2	RESERVED	R/W	0h	Reserved
1	UOEI	R/W	0h	Interrupt enable for DMA under-run or over-run : 0: Interrupt is disable. 1: Interrupt is enable. Over-run in receiving or Under-Run in transmitting.

Table 23-15. INTENSET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DPEI	R/W	0h	Interrupt enable for DMA programming error: 0: Interrupt is disable. 1: Interrupt is enable.

23.5.2.10 INTENCLR Register (Offset = 16h) [reset = 0h]

INTENCLR is shown in [Figure 23-26](#) and described in [Table 23-16](#).

Interrupt Enable Clear Register

Figure 23-26. INTENCLR 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			EOLQ	EOWQ	RESERVED	UOEQ	DPEQ
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED			EOLI	EOWI	RESERVED	UOEI	DPEI
R=0-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-16. INTENCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-13	RESERVED	R=0	0h	Reserved
12	EOLQ	R/W	0h	Interrupt clear for end-of-line condition: 0: No action. 1: Interrupt status gets clear.
11	EOWQ	R/W	0h	Interrupt clear for end-of-line condition: 0: No action. 1: Interrupt status gets clear.
10	RESERVED	R/W	0h	Reserved
9	UOEQ	R/W	0h	Interrupt clear for DMA under-run or over-run : 0: No action. 1: Interrupt status gets clear.
8	DPEQ	R/W	0h	Interrupt clear for DMA programming error: 0: No action. 1: Interrupt status gets clear.
7-5	RESERVED	R=0	0h	Reserved
4	EOLI	R/W	0h	Interrupt clear for end-of-line condition: 0: No action. 1: Interrupt status gets clear.
3	EOWI	R/W	0h	Interrupt clear for end-of-line condition: 0: No action. 1: Interrupt status gets clear.
2	RESERVED	R/W	0h	Reserved
1	UOEI	R/W	0h	Interrupt clear for DMA under-run or over-run : 0: No action. 1: Interrupt status gets clear. Over-run in receiving or Under-Run in transmitting.

Table 23-16. INTENCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DPEI	R/W	0h	Interrupt clear for DMA programming error: 0: No action. 1: Interrupt status gets clear.

23.5.2.11 CHIDESC0 Register (Offset = 20h) [reset = 0h]

CHIDESC0 is shown in [Figure 23-27](#) and described in [Table 23-17](#).

DMA Channel I Descriptor 0 Register

Figure 23-27. CHIDESC0 Register

31	30	29	28	27	26	25	24	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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-17. CHIDESC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Starting address of the DMA Channel I transfer. It must be 64bit aligned.

23.5.2.12 CHIDESC1 Register (Offset = 22h) [reset = 0h]

CHIDESC1 is shown in [Figure 23-28](#) and described in [Table 23-18](#).

DMA Channel I Descriptor 1 Register

Figure 23-28. CHIDESC1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-18. CHIDESC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	LCNT	R/W	0h	Number of lines in a window for DMA Channel I transfer(number of packets in CPPI 4.1 terminology): 0: Line count of 0 (invalid programming) 1: Line count of 1 So on...
15-0	BCNT	R/W	0h	Number of bytes in a line for DMA Channel I transfer(number of bytes in a packet in CPPI 4.1 terminology): 0: Byte count of 0 (invalid programming) 1: Byte count of 1 So on...

23.5.2.13 CHIDESC2 Register (Offset = 24h) [reset = 0h]

CHIDESC2 is shown in [Figure 23-29](#) and described in [Table 23-19](#).

DMA Channel I Descriptor 2 Register

Figure 23-29. CHIDESC2 Register

31	30	29	28	27	26	25	24	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																LOFFSET															
R=0-0h																R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-19. CHIDESC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-0	LOFFSET	R/W	0h	Offset from the current line starting address to the next line starting address for DMA Channel I transfers. It must be 64bit aligned.

23.5.2.14 CHIST0 Register (Offset = 28h) [reset = 0h]

CHIST0 is shown in [Figure 23-30](#) and described in [Table 23-20](#).

DMA Channel I Status 0 Register

Figure 23-30. CHIST0 Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-20. CHIST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	0h	Current address of the DMA transfer.

23.5.2.15 CHIST1 Register (Offset = 2Ah) [reset = 0h]

CHIST1 is shown in [Figure 23-31](#) and described in [Table 23-21](#).

DMA Channel I Status 1 Register

Figure 23-31. CHIST1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-21. CHIST1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	LCNT	R	0h	Current line number.
15-0	BCNT	R	0h	Current byte number.

23.5.2.16 CHIST2 Register (Offset = 2Ch) [reset = 0h]

CHIST2 is shown in [Figure 23-32](#) and described in [Table 23-22](#).

DMA Channel I Status 2 Register

Figure 23-32. CHIST2 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
WM				RESERVED		PEND	ACT
R-0h				R=0-0h		R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-22. CHIST2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R=0	0h	Reserved
7-4	WM	R	0h	Watermark for FIFO block count for DMA Channel I transfer. For RX, this is a recording of the maximum FIFO Block Occupancy ever reached for a continuous transaction. For TX, this is a simple capture of the FIFO Block Emptiness count every clock.
3-2	RESERVED	R=0	0h	Reserved
1	PEND	R	0h	Status of DMA: 0: OK to write a new set of DMA descriptor. 1: Writing of new DMA descriptor is disallowed/ignored.
0	ACT	R	0h	Status of DMA descriptor.t: 0: Descriptor is currently idle. 1: Descriptor is currently active (transferring data). "PENDING" bit is used for descriptor programming allowance, while "ACTIVE" is used for indicating if a descriptor is being in use or not. Software should not use these bit to indicate "end-of-window" transfer condition (use the EOW status/interrupt instead). For RX mode, this bit reflects if any of the 2 pending descriptors on the CBA clock domain is currently running. For TX mode, this bit reflects if any of the 2 pending descriptors on the RPI clock domain is currently running. A proper way to monitor this signal (after loading a descriptor) is to first wait for this ACTIVE signal to go active, then wait for this signal to go inactive. The deassertion of the ACTIVE signal indicates that the descriptor is completed. Another side note is that the "DMA_STATUS" of PCR is a low-level monitoring signal for DMA VBUSP bus activity, while the PENDING, ACTIVE are higher-lever monitoring for descriptor status.

23.5.2.17 CHQDESC0 Register (Offset = 30h) [reset = 0h]

CHQDESC0 is shown in [Figure 23-33](#) and described in [Table 23-23](#).

DMA Channel Q Descriptor 0 Register

Figure 23-33. CHQDESC0 Register

31	30	29	28	27	26	25	24	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																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-23. CHQDESC0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R/W	0h	Starting address of the DMA Channel Q transfer. This must be 64bit aligned.

23.5.2.18 CHQDESC1 Register (Offset = 32h) [reset = 0h]

CHQDESC1 is shown in [Figure 23-34](#) and described in [Table 23-24](#).

DMA Channel Q Descriptor 1 Register

Figure 23-34. CHQDESC1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-24. CHQDESC1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	LCNT	R/W	0h	Number of lines in a window for DMA Channel Q transfer(number of packets in CPPI 4.1 terminology): 0: Line count of 0 (invalid programming) 1: Line count of 1 So on...
15-0	BCNT	R/W	0h	Number of bytes in a line for DMA Channel Q transfer(number of bytes in a packet in CPPI 4.1 terminology): 0: Byte count of 0 (invalid programming) 1: Byte count of 1 So on...

23.5.2.19 CHQDESC2 Register (Offset = 34h) [reset = 0h]

CHQDESC2 is shown in [Figure 23-35](#) and described in [Table 23-25](#).

DMA Channel Q Descriptor 2 Register

Figure 23-35. CHQDESC2 Register

31	30	29	28	27	26	25	24	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																LOFFSET															
R=0-0h																R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-25. CHQDESC2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R=0	0h	Reserved
15-0	LOFFSET	R/W	0h	Offset from the current line starting address to the next line starting address for DMA Channel Q transfers. It must be 64bit aligned.

23.5.2.20 CHQST0 Register (Offset = 38h) [reset = 0h]

CHQST0 is shown in [Figure 23-36](#) and described in [Table 23-26](#).

DMA Channel Q Status 0 Register

Figure 23-36. CHQST0 Register

31	30	29	28	27	26	25	24	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-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-26. CHQST0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ADDR	R	0h	Current address of the DMA transfer.

23.5.2.21 CHQST1 Register (Offset = 3Ah) [reset = 0h]

CHQST1 is shown in [Figure 23-37](#) and described in [Table 23-27](#).

DMA Channel Q Status 1 Register

Figure 23-37. CHQST1 Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-27. CHQST1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	LCNT	R	0h	Current line number.
15-0	BCNT	R	0h	Current byte number.

23.5.2.22 CHQST2 Register (Offset = 3Ch) [reset = 0h]

CHQST2 is shown in [Figure 23-38](#) and described in [Table 23-28](#).

DMA Channel Q Status 2 Register

Figure 23-38. CHQST2 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
WM				RESERVED		PEND	ACT
R=0h				R=0-0h		R=0h	R=0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-28. CHQST2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R=0	0h	Reserved
7-4	WM	R	0h	Watermark for FIFO block count for DMA Channel Q transfer. For RX, this is a recording of the maximum FIFO Block Occupancy ever reached for a continuous transaction. For TX, this is a simple capture of the FIFO Block Emptiness count every clock.
3-2	RESERVED	R=0	0h	Reserved
1	PEND	R	0h	Status of DMA: 0: OK to write a new set of DMA descriptor. 1: Writing of new DMA descriptor is disallowed/ignored.
0	ACT	R	0h	Status of DMA descriptor.t: 0: Descriptor is currently idle. 1: Descriptor is currently active (transferring data). "PENDING" bit is used for descriptor programming allowance, while "ACTIVE" is used for indicating if a descriptor is being in use or not. Software should not use these bit to indicate "end-of-window" transfer condition (use the EOW status/interrupt instead). For RX mode, this bit reflects if any of the 2 pending descriptors on the CBA clock domain is currently running. For TX mode, this bit reflects if any of the 2 pending descriptors on the RPI clock domain is currently running. A proper way to monitor this signal (after loading a descriptor) is to first wait for this ACTIVE signal to go active, then wait for this signal to go inactive. The deassertion of the ACTIVE signal indicates that the descriptor is completed. Another side note is that the "DMA_STATUS" of PCR is a low-level monitoring signal for DMA VBUSP bus activity, while the PENDING, ACTIVE are higher-lever monitoring for descriptor status.

23.5.2.23 GINTEN Register (Offset = 40h) [reset = 0h]

GINTEN is shown in [Figure 23-39](#) and described in [Table 23-29](#).

Global Peripheral Interrupt Enable Register

Figure 23-39. GINTEN 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							GINTEN
R=0-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-29. GINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R=0	0h	Reserved
0	GINTEN	R/W	0h	0 = uPP does not generate interrupt. 1= uPP generates interrupt to if interrupt flag gets set.

23.5.2.24 GINTFLG Register (Offset = 42h) [reset = 0h]

GINTFLG is shown in [Figure 23-40](#) and described in [Table 23-30](#).

Global Peripheral Interrupt Flag Register

Figure 23-40. GINTFLG 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							GINTFLG
R=0-0h							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-30. GINTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R=0	0h	Reserved
0	GINTFLG	R	0h	0: No interrupt has been generated. 1: Interrupt has been generated.

23.5.2.25 GINTCLR Register (Offset = 44h) [reset = 0h]

GINTCLR is shown in [Figure 23-41](#) and described in [Table 23-31](#).

Global Peripheral Interrupt Clear Register

Figure 23-41. GINTCLR 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							GINTCLR
R=0-0h							R=0/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-31. GINTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R=0	0h	Reserved
0	GINTCLR	R=0/W=1	0h	Write '1' to this clears the flag in GINTFR. Read always returns '0'.

23.5.2.26 DLYCTL Register (Offset = 46h) [reset = 1h]

DLYCTL is shown in [Figure 23-42](#) and described in [Table 23-32](#).

IO clock data skew control Register

Figure 23-42. DLYCTL 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					DLYCTL		DLYDIS
R=0-0h					R/W-0h		R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 23-32. DLYCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R=0	0h	Reserved
2-1	DLYCTL	R/W	0h	Controls the delay on the input signals for uPP. 00: Data and control pins have 4 cycle dealy, clock pins have 2 cycle delay. 01: Data and control pins have 6 cycle dealy, clock pins have 2 cycle delay. 10: Data and control pins have 9 cycle dealy, clock pins have 2 cycle delay. 11: Data and control pins have 14 cycle dealy, clock pins have 2 cycle delay.
0	DLYDIS	R/W	1h	0: Dealy on pins are controlled by setting in DLYCTL field. 1: No extra dealy on pins.

External Memory Interface (EMIF)

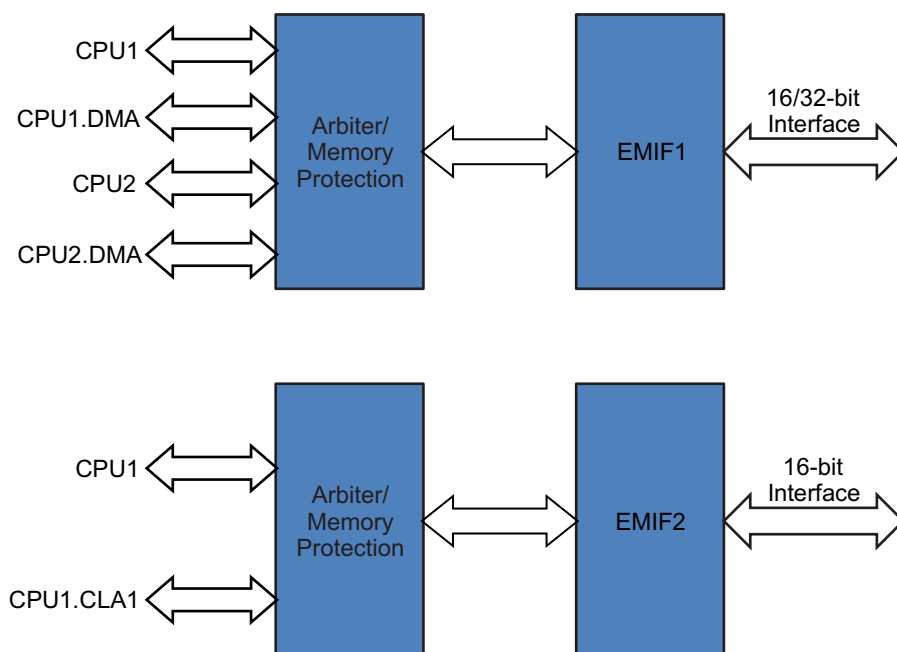
This chapter describes the external memory interface (EMIF).

Topic	Page
24.1 Introduction	2319
24.2 Configuring Device Pins	2321
24.3 EMIF Module Architecture	2321
24.4 Example Configuration	2354
24.5 Registers	2362

24.1 Introduction

This device supports dual-core architecture; in order to have a dedicated EMIF for each CPU subsystem, the device supports two EMIF modules — EMIF1 and EMIF2. Both modules are exactly the same with the same feature set, but have different address/data sizes. EMIF1 is shared between the CPU1 and CPU2 subsystem, whereas EMIF2 is dedicated to the CPU1 subsystem. [Figure 24-1](#) represents the two modules.

Figure 24-1. EMIF Module Overview



[Table 24-1](#) gives the configuration for two EMIF modules.

Table 24-1. Configuration for EMIF1 and EMIF2 Modules

	EMIF1	EMIF2
176-Pin Package	Yes	NA
337- Pin Package	Yes	Yes
Max Data Width	32	16
Max Address Width	22 (Some of EMIF1 pins are muxed with each other. Please refer to the IO mux section for exact usage)	12
SDRAM CSx Support	1 (CS0)	1(CS0)
ASRAM CSx Support	3 (CS2/CS3/CS4)	1(CS2)

NOTE: Subsequent sections in this chapter will provide the details on generic EMIF modules unless otherwise specified. Pin names are used from EMIF1 to define the functionality.

NOTE: On this device, if EMIF1 is chosen to have a 32-bit data width, EMIF2 cannot be used because EMIF2 data pins are muxed with EMIF1 MSB data pins.

24.1.1 Purpose of the Peripheral

This EMIF memory controller is compliant with the JESD21-C SDR SDRAM memories utilizing a 32-bit/16-bit data bus. The purpose of this EMIF is to provide a means for the CPU to connect to a variety of external devices including:

- Single data rate (SDR) SDRAM
- Asynchronous devices including NOR Flash and SRAM

A common use for the EMIF is to interface with both a flash device and an SDRAM device simultaneously. [Section 24.4](#) contains an example of operating the in this configuration.

24.1.2 Features

The EMIF controller includes many features to enhance the ease and flexibility of connecting to the external SDR SDRAM and asynchronous devices.

24.1.2.1 Asynchronous Memory Support

The EMIF controller supports asynchronous:

- SRAM memories
- NOR Flash memories

There is an external wait input that allows slower asynchronous memories to extend the memory access. The EMIF module supports more than one chip select (enable). Each chip select has the following individually programmable attributes:

- Data bus width
- Read cycle timings: setup, hold, strobe
- Write cycle timings: setup, hold, strobe
- Bus turnaround time
- Extended wait option with programmable timeout
- Select strobe option

24.1.2.2 Synchronous DRAM Memory Support

The EMIF module supports 16-bit/32-bit SDRAM in addition to the asynchronous memories listed in [Section 24.1.2.1](#). It has a single SDRAM chip select. SDRAM configurations that are supported are:

- One, two and four bank SDRAM devices
- Devices with eight, nine, ten, and eleven column address
- CAS latency of two or three clock cycles
- 16-bit/32-bit data bus width
- 3.3V LVCMOS interface

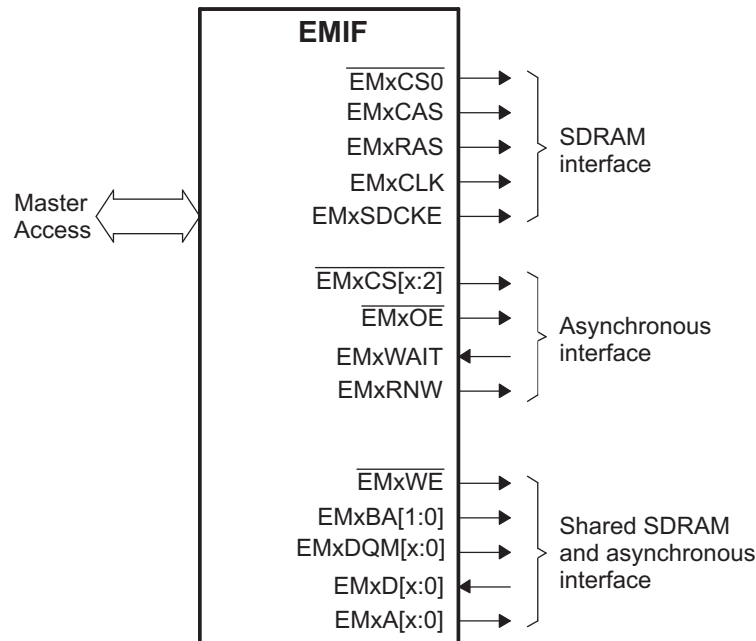
Additionally, the EMIF supports placing the SDRAM in self-refresh and power-down modes. The self-refresh mode allows the SDRAM to be put in a low-power state while still retaining memory contents, since the SDRAM will continue to refresh itself even without clocks from the microcontroller. The power-down mode achieves even lower power, except the microcontroller must periodically wake up and issue refreshes if data retention is required.

Note that the EMIF module does not support mobile SDRAM devices.

24.1.3 Functional Block Diagram

[Figure 24-2](#) illustrates the connections between the EMIF and its internal requesters, along with the external EMIF pins. [Section 24.3.2](#) contains a description of the entities internal to the MCU that can send requests to the EMIF, along with their prioritization. [Section 24.3.3](#) describes the EMIF external pins and summarizes their purpose when interfacing with the SDRAM and asynchronous devices.

Figure 24-2. EMIF Functional Block Diagram



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 EMIF Module Architecture

This section provides details about the architecture and operation of the EMIF. Both the SDRAM and asynchronous interface are covered, along with other system-related issues such as clock control.

24.3.1 EMIF Clock Control

The EMIF clock is output on the EMxCLK pin and should be used when interfacing to external SDRAM devices. The EMIF module gets the PLLSYSCLK clock domain as the input. The user can choose to run the EMIF at PLLSYSCLK/1 or PLLSYSCLK/2 clock frequency by configuring the EMIFxCLKDIV field in the PERCLKDIVSEL register in the *Clock Control* module.

24.3.2 EMIF Requests

Different sources within the MCU can make requests to the EMIF. These requests consist of accesses to the SDRAM memory, the asynchronous memory, and the EMIF registers. The EMIF can process only one request at a time. Therefore, a high performance master arbitration block exists within the MCU to provide prioritized requests from the different sources to the EMIF. The sources are:

- CPU1
- CPU1.DMA
- CPU2
- CPU2.DMA

If a request is submitted from two or more sources simultaneously, the master arbitration block will forward the highest priority request to the EMIF first. Upon completion of a request, the master arbitration block again evaluates the pending requests and forwards the highest priority pending request to the EMIF.

The master arbitration block always allows RD access from any of the masters. But for WR access (or execute access), the master arbitration block only allows access of masters from a CPU subsystem which grabs master ownership of the EMIF module based on the configuration in the EMIF1MSEL register in the *Memory Controller* module. Please note that only the EMIF1 has access from both the CPU subsystems; hence the concept of grab-semaphore is applicable for the EMIF1 only.

When the EMIF receives a request, it may or may not be immediately processed. In some cases, the EMIF will perform one or more auto refresh cycles before processing the request. For details on the EMIF's internal arbitration between performing requests and performing auto refresh cycles, see [Section 24.3.13](#).

24.3.3 EMIF Signal Descriptions

This section describes the function of each of the EMIF signals.

Table 24-2. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories

Pins(s)	I/O	Description
EM1D[x:0]	I/O	EMIF data bus.
EM1A[x:0]	O	EMIF address bus. When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in Table 24-14 . EM1A[10] is also used during the PRE command to select which banks to deactivate. When interfacing to an asynchronous device, these pins are used in conjunction with the EM1BA pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in Section 24.3.6.1 .
EM1BA[1:0]	O	EMIF bank address. When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in Table 24-14 . When interfacing to an asynchronous device, these pins are used in conjunction with the EM1A pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in Section 24.3.6.1 .
EM1DQM[x:0]	O	Active-low byte enables. When interfacing to SDRAM, these pins are connected to the DQM pins of the SDRAM to individually enable/disable each of the bytes in a data access. When interfacing to an asynchronous device, these pins are connected to byte enables. See Section 24.3.6 for details.
EM1WE	O	Active-low write enable. When interfacing to SDRAM, this pin is connected to the nWE pin of the SDRAM and is used to send commands to the device. When interfacing to an asynchronous device, this pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle.

Table 24-3. EMIF Pins Specific to SDRAM

Pin(s)	I/O	Description
EM1CS0	O	Active-low chip enable pin for SDRAM devices. This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, EMIF keeps this SDRAM chip select active, even if EMIF is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous access.
EM1RAS	O	Active-low row address strobe pin. This pin is connected to the nRAS pin of the attached SDRAM device and is used for sending commands to the device.
EM1CAS	O	Active-low column address strobe pin. This pin is connected to the nCAS pin of the attached SDRAM device and is used for sending commands to the device.

Table 24-3. EMIF Pins Specific to SDRAM (continued)

Pin(s)	I/O	Description
EM1SDCKE	O	Clock enable pin. This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self-refresh mode. See Section 24.3.5.7 for details.
EM1CLK	O	SDRAM clock pin. This pin is connected to the CLK pin of the attached SDRAM device. See Section 24.3.1 for details on the clock signal.

Table 24-4. EMIF Pins Specific to Asynchronous Memory

Pin(s)	I/O	Description
EM1CS[4:2]	O	Active-low chip enable pins for asynchronous devices. These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory.
EM1WAIT	I	Wait input with programmable polarity. A connected asynchronous device can extend the strobe period of an access cycle by asserting the EM1WAIT input to EMIF as described in Section 24.3.6.6 . To enable this functionality, the EW bit in the asynchronous 1 configuration register (ASYNC_CS2_CFG) must be set to 1. In addition, the WPO bit in ASYNC_CS2_CFG must be configured to define the polarity of the EM1WAIT pin.
EM1TOE	O	Active-low pin enable for asynchronous devices. This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EM1RNW	O	EMIF asynchronous read/write control. This pin stays high during reads and stays low during writes (same duration as CS).

24.3.4 EMIF Signal Multiplexing Control

Several EMIF signals are multiplexed with other functions on this microcontroller. Please refer to the multiplexing section of the *GPIO* chapter for more information on how to enable the output of these EMIF signals.

24.3.5 SDRAM Controller and Interface

The EMIF controller can gluelessly interface to most standard SDR SDRAM devices and supports such features as self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to Interface and properly configure the EMIF to perform read and write operations to externally connected SDR SDRAM devices. Also, [Section 24.4](#) provides a detailed example of interfacing the EMIF to a common SDRAM device.

24.3.5.1 SDRAM Commands

The EMIF controller supports the SDRAM commands described in [Table 24-5](#). [Table 24-6](#) shows the truth table for the SDRAM commands, and an example timing waveform of the PRE command is shown in [Figure 24-3](#). EM1A[10] is pulled low in this example to deactivate only the bank specified by the EM1BA pins.

Table 24-5. EMIF SDRAM Commands

Command	Function
PRE	Precharge. Depending on the value of EM1A[10], the PRE command either deactivates the open row in all banks (EM1A[10] = 1) or only the bank specified by the EM1BA[1:0] pins (EM1A[10] = 0).
ACTV	Activate. The ACTV command activates the selected row in a particular bank for the current access.

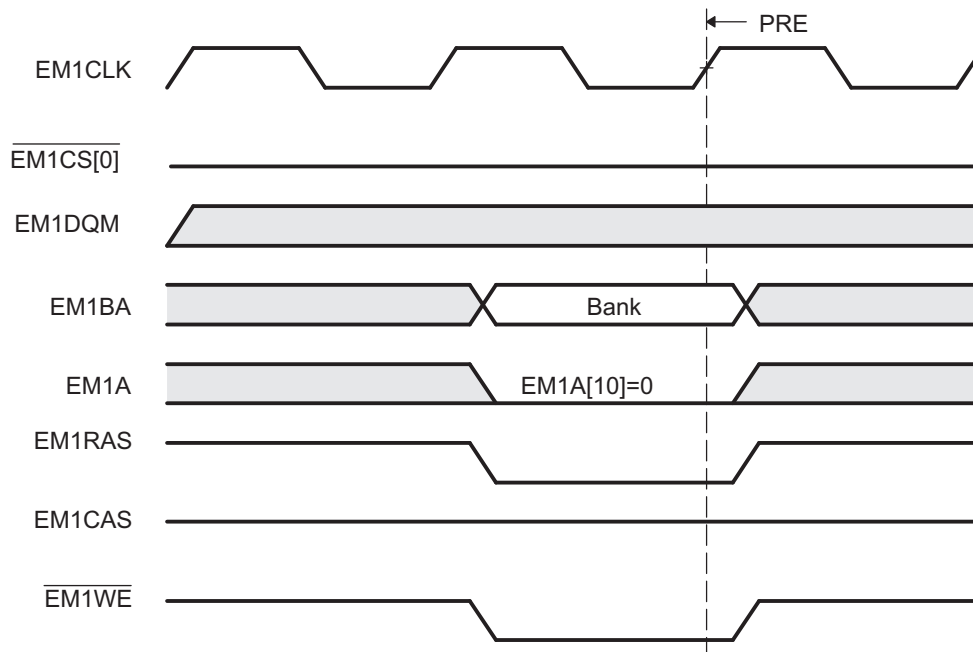
Table 24-5. EMIF SDRAM Commands (continued)

Command	Function
READ	Read. The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EM1A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
WRT	Write. The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EM1A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
BT	Burst terminate. The BT command is used to truncate the current read or write burst request.
LMR	Load mode register. The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in Section 24.3.5.4 .
REFR	Auto refresh. The REFR command signals the SDRAM to perform an auto refresh according to its internal address.
SLFR	Self refresh. The self-refresh command places the SDRAM into self-refresh mode, during which it provides its own clock signal and auto refresh cycles.
NOP	No operation. The NOP command is issued during all cycles in which one of the above commands is not issued.

Table 24-6. Truth Table for SDRAM Commands

SDRAM Pins:	CKE	nCS	nRAS	nCAS	nWE	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIF Pins:	EM1SDCKE	EM1CS[0]	EM1RAS	EM1CAS	EM1WE	EM1BA[1:0]	EM1A[12:11]	EM1A[10]	EM1A[9:0]
PRE	H	L	L	H	L	Bank/X	X	L/H	X
ACTV	H	L	L	H	H	Bank	Row	Row	Row
READ	H	L	H	L	H	Bank	Column	L	Column
WRT	H	L	H	L	L	Bank	Column	L	Column
BT	H	L	H	H	L	X	X	X	X
LMR	H	L	L	L	L	X	Mode	Mode	Mode
REFR	H	L	L	L	H	X	X	X	X
SLFR	L	L	L	L	H	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X

Figure 24-3. Timing Waveform of SDRAM PRE Command



24.3.5.2 Interfacing to SDRAM

The EMIF supports a glueless interface to SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9, 10, or 11.
- The number of row address bits is 13, 14, 15, or 16.
- The number of internal banks is 1, 2, or 4.

Figure 24-4 shows an interface between the EMIF and a 2M x 16 x 4 bank SDRAM device, and Figure 24-5 shows an interface between the EMIF and a 512K x 16 x 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 24-7 for list of commonly-supported SDRAM devices and the required connections for the address pins.

Figure 24-4. EMIF to 2M x 16 x 4 bank SDRAM Interface

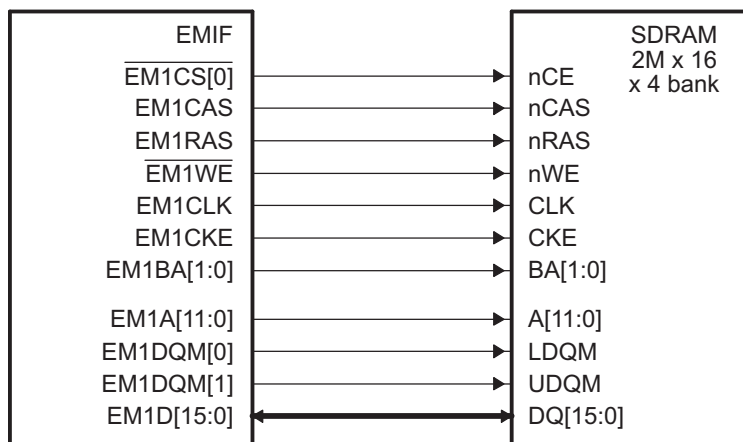
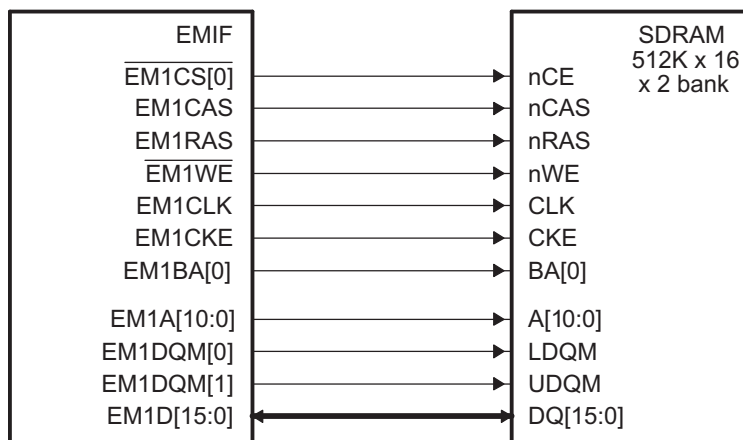


Figure 24-5. EMIF to 512K x 16 x 2 bank SDRAM Interface

Table 24-7. 16-bit EMIF Address Pin Connections

SDRAM Size	Width	Banks	Device	Address Pins
16M bits	x16	2	SDRAM	A[10:0]
			EMIF	EM1A[10:0]
64M bits	x16	4	SDRAM	A[11:0]
			EMIF	EM1A[11:0]
128M bits	x16	4	SDRAM	A[11:0]
			EMIF	EM1A[11:0]
256M bits	x16	4	SDRAM	A[12:0]
			EMIF	EM1A[12:0]
512M bits	x16	4	SDRAM	A[12:0]
			EMIF	EM1A[12:0]

24.3.5.3 SDRAM Configuration Registers

The operation of the EMIF's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but [Section 24.5](#) should be referred for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the four such configuration registers, along with a description of each of their programmable fields.

NOTE: Writing to any of the fields: NM, CL, IBANK, and PAGESIZE in the SDRAM configuration register (SDRAM_CR) causes the EMIF to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in [Section 24.3.5.4](#).

Table 24-8. Description of the SDRAM Configuration Register (SDRAM_CR)

Parameter	Description
SR	This bit controls entering and exiting of the self-refresh mode
PD	This bit controls entering and exiting of the power-down mode. If both SR and PD bits are set, the EMIF will go into self-refresh mode.
PDWR	Perform refreshes during power Down. Writing a 1 to this bit will cause the EMIF to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set. This bit should be set along with PD when entering power-down mode.
NM	Narrow Mode. This bit defines the width of the data bus between the EMIF and the attached SDRAM device. When set to 1, the data bus is set to 16-bits. When set to 0, the data bus is set to 32-bits.
CL	CAS latency. This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device via the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in Section 24.3.5.4 . Only, values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and should be written to this field. While updating the CL field, BIT11_9LOCK bit field must be set to '1' simultaneously.
IBANK	Number of Internal SDRAM Banks. This field defines the number of banks inside the attached SDRAM devices in the following way: <ul style="list-style-type: none"> When IBANK = 0, 1 internal bank is used When IBANK = 1h, 2 internal banks are used When IBANK = 2h, 4 internal banks are used This field value affects the mapping of logical addresses to the SDRAM row, column, and bank addresses. See Section 24.3.5.11 for details.
PAGESIZE	Page Size. This field defines the internal page size of the attached SDRAM devices in the following way: <ul style="list-style-type: none"> When PAGESIZE = 0, 256-word pages are used When PAGESIZE = 1h, 512-word pages are used When PAGESIZE = 2h, 1024-word pages are used When PAGESIZE = 3h, 2048-word pages are used This field value affects the mapping of logical addresses to the SDRAM row, column, and bank addresses. See Section 24.3.5.11 for details.

Table 24-9. Description of the SDRAM Refresh Control Register (SDRAM_RCR)

Parameter	Description
RR	Refresh Rate. This field controls the rate at which attached SDRAM devices will be refreshed. The following equation can be used to determine the required value of RR for an SDRAM device: <ul style="list-style-type: none"> $RR = f_{EM1CLK} / (\text{Required SDRAM Refresh Rate})$ More information about the operation of the SDRAM refresh controller can be found in Section 24.3.5.6 .

Table 24-10. Description of the SDRAM Timing Register (SDRAM_TR)

Parameter	Description
T_RFC	SDRAM Timing Parameters. These fields configure the EMIF to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIF to avoid violating SDRAM timing constraints and to more efficiently schedule its operations. More details about each of these parameters can be found in the SDRAM_TR register description. These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM data sheet.
T_RP	
T_RCD	
T_WR	
T_RAS	
T_RC	
T_RRD	

Table 24-11. Description of the SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)

Parameter	Description
T_XS	Self Refresh Exit Parameter. The T_XS field of this register informs the EMIF about the minimum number of EM1CLK cycles required between exiting self-refresh and issuing any command. This parameter should be set to satisfy the t_{XSR} value for the attached SDRAM device.

24.3.5.4 SDRAM Auto-Initialization Sequence

The EMIF automatically performs an SDRAM initialization sequence, regardless of whether it is interfaced to an SDRAM device, when either of the following two events occur:

- The EMIF comes out of reset. No memory accesses to the SDRAM and asynchronous interfaces are performed until this auto-initialization is complete.
- A write is performed to any of the three least significant bytes of the SDRAM configuration register (SDRAM_CR)

An SDRAM initialization sequence consists of the following steps:

1. If the initialization sequence is activated by a write to SDRAM_CR, and if any of the SDRAM banks are open, the EMIF issues a PRE command with EM1A[10] held high to indicate all banks. This is done so that the maximum ACTV to PRE timing for an SDRAM is not violated.
2. The EMIF drives EM1SDCKE high and begins continuously issuing NOP commands until eight SDRAM refresh intervals have elapsed. An SDRAM refresh interval is equal to the value of the RR field of the SDRAM refresh control register (SDRAM_RCR), divided by the frequency of EM1CLK (RR/f_{EM1CLK}). This step is used to avoid violating the power-up constraint of most SDRAM devices that requires 200 μ s (sometimes 100 μ s) between receiving stable Vdd and CLK and the issuing of a PRE command. Depending on the frequency of EM1CLK, this step may or may not be sufficient to avoid violating the SDRAM constraint. See [Section 24.3.5.5](#) for more information.
3. After the refresh intervals have elapsed, the EMIF issues a PRE command with EM1A[10] held high to indicate all banks.
4. The EMIF issues eight AUTO REFRESH commands.
5. The EMIF issues the LMR command with the EM1A[9:0] pins set as described in [Table 24-12](#).
6. Finally, the EMIF performs a refresh cycle, which consists of the following steps:
 - (a) Issuing a PRE command with EM1A[10] held high if any banks are open
 - (b) Issuing an REF command

Table 24-12. SDRAM LOAD MODE REGISTER Command

EM1A[9:7]	EM1A[6:4]	EM1A[3]	EM1A[2:0]
0 (Write bursts are of the programmed burst length in EM1A[2:0])	These bits control the CAS latency of the SDRAM and are set according to CL field in the SDRAM configuration register (SDRAM_CR) as follows: <ul style="list-style-type: none"> If CL = 2, EM1A[6:4] = 2h (CAS latency = 2) If CL = 3, EM1A[6:4] = 3h (CAS latency = 3) 	0 (Sequential Burst Type. Interleaved Burst Type not supported)	These bits control the burst length of the SDRAM and are set according to the NM field in the SDRAM configuration register (SDRAM_CR) as follows: <ul style="list-style-type: none"> If NM = 0, EM1A[2:0] = 2h (Burst Length = 4) If NM = 1, EM1A[2:0] = 3h (Burst Length = 8)

24.3.5.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although the EMIF automatically performs the SDRAM initialization sequence described in [Section 24.3.5.4](#) when coming out of reset, it is recommended to follow one of the procedures listed below before performing any EMIF memory requests. Procedure A should be followed if it is determined that the SDRAM power-up constraint was not violated during the SDRAM auto-initialization sequence detailed in [Section 24.3.5.4](#) on coming out of Reset. The SDRAM power-up constraint specifies that 200 μ s (sometimes 100 μ s) should exist between receiving stable V_{dd} and CLK and the issuing of a PRE command. Procedure B should be followed if the SDRAM power-up constraint was violated. The 200 μ s (100 μ s) SDRAM power-up constraint will be violated if the frequency of EM1CLK is greater than 50 MHz (100 MHz for 100 μ s SDRAM power-up constraint) during SDRAM Auto-Initialization Sequence. Procedure B should be followed if there is any doubt that the power-up constraint was not met.

Procedure A — Following is the procedure to be followed if the SDRAM power-up constraint was NOT violated:

1. Place the SDRAM into self-refresh mode by setting the SR bit of SDRAM_CR to 1. The SDRAM should be placed into self-refresh mode when changing the frequency of the EM1CLK to avoid incurring the 200 μ s power-up constraint again.
2. Configure the desired EM1CLK clock frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device data manual.
3. Remove the SDRAM from self-refresh Mode by clearing the SR bit of the SDRAM_CR to 0.
4. Program SDRAM_TR and SDR_EXT_TMNG to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM data sheet.
5. Program the RR field of SDRAM_RCR to match that of the attached device's refresh interval. See [Section 24.3.5.6.1](#) details on determining the appropriate value.
6. Program the SDRAM_CR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 24.3.5.4](#) to be re-run. This second initialization generally takes much less time due to the increased frequency of EM1CLK.

Procedure B — Following is the procedure to be followed if the SDRAM power-up constraint was violated:

1. Configure the desired EM1CLK clock frequency. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device data manual.
2. Program SDRAM_TR and SDR_EXT_TMNG to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM data sheet.
3. Program the RR field of the SDRAM_RCR such that the following equation is satisfied: $(RR \times 8)/(f_{EM1CLK}) > 200 \mu s$ (sometimes 100 μ s). For example, an EM1CLK frequency of 100 MHz would require setting RR to 2501 (9C5h) or higher to meet a 200 μ s constraint.
4. Program the SDRAM_CR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 24.3.5.4](#) to be re-run with the new value of RR.

5. Perform a read from the SDRAM to assure that step 5 of this procedure will occur after the initialization process has completed. Alternatively, wait for 200 μ s instead of performing a read.
6. Finally, program the RR field to match that of the attached device's refresh interval. See [Section 24.3.5.6.1](#) details on determining the appropriate value.

After following the above procedure, the EMIF is ready to perform accesses to the attached SDRAM device.

24.3.5.6 EMIF Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIF can meet this constraint by performing auto refresh cycles at or above this required rate. An auto-refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIF of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRAM_RCR) must be programmed. The EMIF will use this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto-refresh cycles cannot be disabled, even if the EMIF is not interfaced with an SDRAM. The remainder of this section details the EMIF's refresh scheme and provides an example for determining the appropriate value to place in the RR field of the SDRAM_RCR.

The two counters used to perform auto-refresh cycles are a 13-bit refresh interval counter and a 4-bit refresh backlog counter. At reset and upon writing to the RR field, the refresh interval counter is loaded with the value from RR field and begins decrementing, by one, each EMIF clock cycle. When the refresh interval counter reaches zero, the following actions occur:

- The refresh interval counter is reloaded with the value from the RR field and restarts decrementing.
- The 4-bit refresh backlog counter increments unless it has already reached its maximum value.

The refresh backlog counter records the number of auto refresh cycles that the EMIF currently has outstanding. This counter is decremented by one each time an auto refresh cycle is performed and incremented by one each time the refresh interval counter expires. The refresh backlog counter saturates at the values of 0000b and 1111b. The EMIF uses the refresh backlog counter to determine the urgency with which an auto refresh cycle should be performed. The four levels of urgency are described in [Table 24-13](#). This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

Table 24-13. Refresh Urgency Levels

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIF has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIF has no requests pending, regardless of whether any SDRAM banks are open.
Refresh Need	8-11	An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending.
Refresh Must	12-15	Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIF can begin servicing any new read or write requests.

24.3.5.6.1 Determining the Appropriate Value for the RR Field

The value that should be programmed into the RR field of the SDRAM_RCR can be calculated by using the frequency of the EM1CLK signal (f_{EM1CLK}) and the required refresh rate of the SDRAM ($f_{Refresh}$). The following formula can be used:

$$RR = f_{EM1CLK} / f_{Refresh}$$

The SDRAM data sheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can therefore be calculated by dividing the number of required cycles per time interval (n_{cycles}) by the time interval given in the data manual ($t_{Refresh\ Period}$):

$$f_{Refresh} = n_{cycles} / t_{Refresh\ Period}$$

Combining these formulas, the value that should be programmed into the RR field can be computed as:

$$RR = f_{EM1CLK} \times t_{Refresh\ Period} / n_{cycles}$$

The following example illustrates calculating the value of RR. Given that:

- $f_{EM1CLK} = 100\text{ MHz}$ (frequency of EMIF clock)
- $t_{Refresh\ Period} = 64\text{ ms}$ (required refresh interval of the SDRAM)
- $n_{cycles} = 8192$ (number of cycles in a refresh interval for the SDRAM)

RR can be calculated as:

$$RR = 100\text{ MHz} \times 64\text{ ms} / 8192$$

$$RR = 781.25$$

$$RR = 782\text{ cycles} = 30\text{Eh cycles}$$

24.3.5.7 Self-Refresh Mode

The EMIF can be programmed to enter the self-refresh state by setting the SR bit of SDRAM_CR to 1. This will cause the EMIF to issue the SLFR command after completing any outstanding SDRAM access requests and clearing the refresh backlog counter by performing one or more auto refresh cycles. This places the attached SDRAM device into self-refresh mode in which it consumes a minimal amount of power while performing its own refresh cycles.

While in the self-refresh state, the EMIF continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIF will not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, in order to prevent floating inputs on the data bus. More information about data bus parking can be found in [Section 24.3.7](#).

The EMIF will exit from the self-refresh state if either of the following events occur:

- The SR bit of SDRAM_CR is cleared to 0.
- An SDRAM accesses is requested.

The EMIF exits from the self-refresh state by driving EM1SDCKE high and performing an auto refresh cycle.

The attached SDRAM device should also be placed into self-refresh mode when changing the frequency of EM1CLK. If the frequency of EM1CLK changes while the SDRAM is not in self-refresh mode, Procedure B in [Section 24.3.5.5](#) should be followed to reinitialize the device.

24.3.5.8 Power Down Mode

To support low-power modes, the EMIF can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in the SDRAM configuration register (SDRAM_CR). When this bit is set, the EMIF will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog (if there is one) has been cleared. At this point the EMIF will enter the power-down state. Upon entering this state, the EMIF will issue a POWER DOWN command (same as a NOP command but driving the EM1SDCKE low on the same cycle). The EMIF then maintains the EM1SDCKE low until it exits the power-down state.

Since the EMIF services the refresh backlog before it enters the power-down state, all internal banks of the SDRAM are closed (precharged) prior to issuing the POWER DOWN command. Therefore, the EMIF only supports precharge power-down. The EMIF does not support active power-down, where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power-down state, the EMIF services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power-down state upon completion.

The PDWR bit in the SDRAM_CR indicates whether the EMIF should perform refreshes in power-down state. If the PDWR bit is set, the EMIF exits the power-down state every time the Refresh Must level is set, performs AUTO REFRESH commands to the SDRAM, and returns back to the power-down state. This evenly distributes the refreshes to the SDRAM in power-down state. If the PDWR bit is not set, the EMIF does not perform any refreshes to the SDRAM. Therefore, the data integrity of the SDRAM is not assured upon power-down exit if the PDWR bit is not set.

If the PD bit is cleared while in the power-down state, the EMIF will come out of the power-down state. The EMIF:

- Drives EM1SDCKE high
- Enters its idle state

24.3.5.9 SDRAM Read Operation

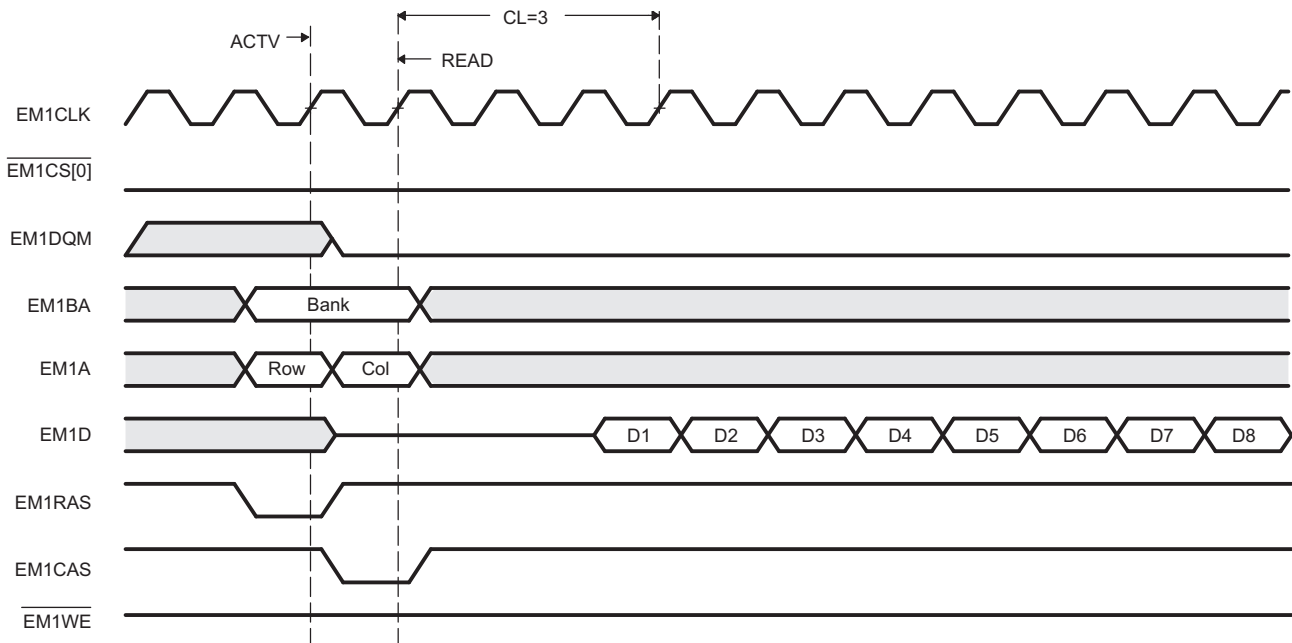
When the EMIF receives a read request to the SDRAM from one of the requesters listed in [Section 24.3.2](#), it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a READ command while specifying the desired bank and column address. EM1A[10] is held low during the READ command to avoid auto-precharging. The READ command signals the SDRAM device to start bursting data from the specified address while EMIF issues NOP commands. Following a READ command, the CL field of the SDRAM configuration register (SDRAM_CR) defines how many delay cycles will be present before the read data appears on the data bus. This is referred to as the CAS latency.

[Figure 24-6](#) shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. When the EMIF SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDRAM_CR) to 1, a burst size of eight is used. [Figure 24-6](#) shows a burst size of eight.

The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIF can truncate the burst in three ways:

- By issuing another READ to the same page in the same bank.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

Figure 24-6. Timing Waveform for Basic SDRAM Read Operation



Several other pins are also active during a read access. The EM1DQM[x:0] pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIF pins during each command can be found in [Table 24-6](#).

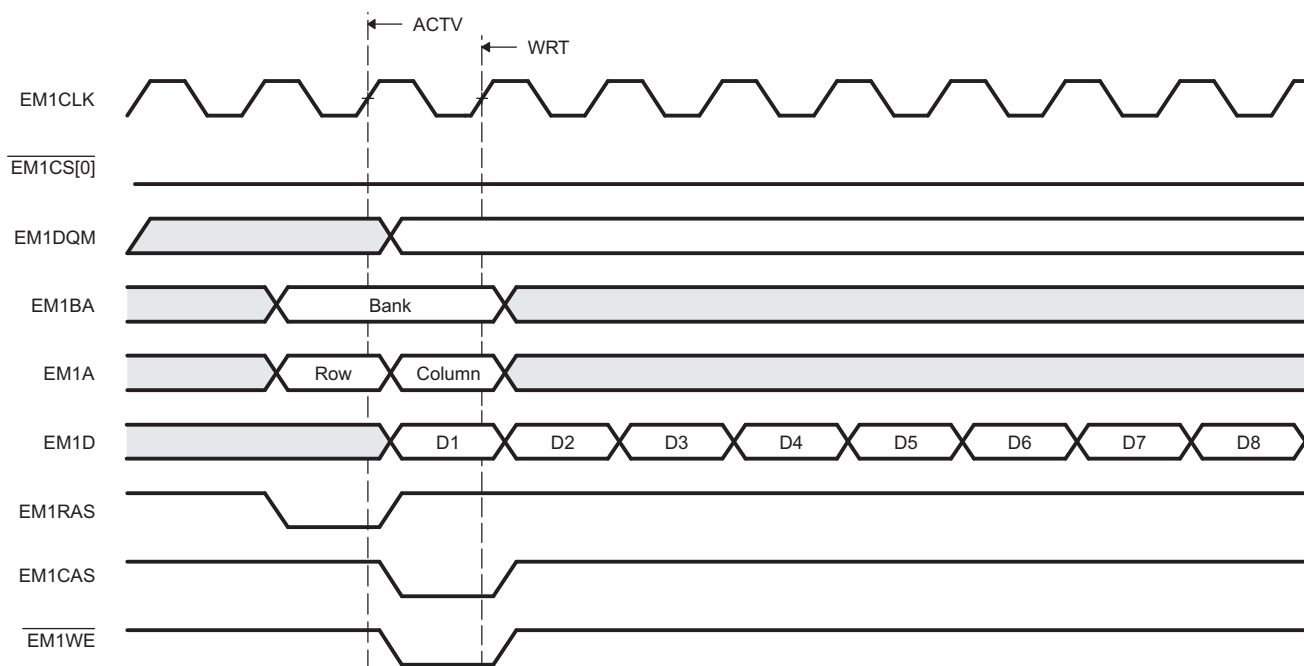
The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDRAM_TR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM data manual. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. Refer to the register description of SDRAM_TR in the SDTIMER register for more details on the various timing parameters.

24.3.5.10 SDRAM Write Operations

When the EMIF receives a write request to SDRAM from one of the requesters listed in [Section 24.3.2](#), it performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a WRT command while specifying the desired bank and column address. EM1A[10] is held low during the WRT command to avoid auto-precharging. The WRT command signals the SDRAM device to start writing a burst of data to the specified address while the EMIF issues NOP commands. The associated write data will be placed on the data bus in the cycle concurrent with the WRT command and with subsequent burst continuation NOP commands.

[Figure 24-7](#) shows the signal waveforms for a basic SDRAM write operation in which a burst of data is read from a single page. When the EMIF SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDRAM_CR) to 1, a burst size of eight is used. [Figure 24-7](#) shows a burst size of eight.

Figure 24-7. Timing Waveform for Basic SDRAM Write Operation



The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIF can truncate the burst in three ways:

- By issuing another WRT to the same page
- By issuing a PRE command in order to prepare for accessing a different page of the same bank
- By issuing a BT command in order to prepare for accessing a page in a different bank

Several other pins are also active during a write access. The EM1DQM[x:0] pins are driven to select which bytes of the data word will be written to the SDRAM device. They are also used to mask out entire undesired data words during a burst access. The state of the other EMIF pins during each command can be found in [Table 24-6](#).

The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDRAM_TR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM data sheet. The EMIF uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. Refer to the register description of SDRAM_TR in the SDTIMR register for more details on the various timing parameters.

24.3.5.11 Mapping from Logical Address to EMIF Pins

When the EMIF receives an SDRAM access request, it must convert the address of the access into the appropriate signals to send to the SDRAM device. The details of this address mapping are shown in [Table 24-14](#) for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDRAM_CR), the EMIF determines which bits of the logical address will be mapped to the SDRAM row, column, and bank addresses.

As the logical address is incremented by one halfword (16-bit operation), the column address is likewise incremented by one until a page boundary is reached. When the logical address increments across a page boundary, the EMIF moves into the same page in the next bank of the attached device by incrementing the bank address EM1BA and resetting the column address. The page in the previous bank is left open until it is necessary to close it. This method of traversal through the SDRAM banks helps maximize the number of open banks inside of the SDRAM and results in an efficient use of the device. There is no limitation on the number of banks that can be open at one time, but only one page within a bank can be open at a time.

The EMIF uses the EM1DQM[1:0] pins during a WRT command to mask out selected bytes or entire words. The EM1DQM[1:0] pins are always low during a READ command.

Table 24-14. Mapping from Logical Address to EMIF Pins for 32-bit SDRAM

IBANK	PAGESIZE	Logical Address												8:1	0
		31:27	26	25	24	23	22	21:14	13	12	11	10	9		
0	0	-												Col Address	EM1DQM[0]/EM1DQM[1]
1	0	-												Col Address	EM1DQM[0]/EM1DQM[1]
2	0	-												Col Address	EM1DQM[0]/EM1DQM[1]
0	1	-												Col Address	EM1DQM[0]
1	1	-												Col Address	EM1DQM[0]/EM1DQM[1]
2	1	-												Col Address	EM1DQM[0]/EM1DQM[1]
0	2	-												Col Address	EM1DQM[0]/EM1DQM[1]
1	2	-												Col Address	EM1DQM[0]/EM1DQM[1]
2	2	-												Col Address	EM1DQM[0]/EM1DQM[1]
0	3	-												Col Address	EM1DQM[0]/EM1DQM[1]
1	3	-												Col Address	EM1DQM[0]/EM1DQM[1]
2	3	-												Col Address	EM1DQM[0]/EM1DQM[1]

Table 24-15. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM

IBANK	PAGESIZE	Logical Address												8	7:0
		31:26	25	24	23	22	21	20:13	12	11	10	9	8		
0	0	-												Col Address	
1	0	-												Col Address	
2	0	-												Col Address	
0	1	-												Col Address	
1	1	-												Col Address	
2	1	-												Col Address	
0	2	-												Col Address	
1	2	-												Col Address	
2	2	-												Col Address	
0	3	-												Col Address	
1	3	-												Col Address	
2	3	-												Col Address	

NOTE: The upper bit of the row address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

24.3.6 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including NOR Flash and SRAM. It can be operated in two major modes (see [Table 24-16](#)):

- Normal Mode
- Select Strobe Mode

Table 24-16. Normal Mode vs. Select Strobe Mode

Mode	Function of EM1DQM pins	Operation of $\overline{\text{EM1CS}}[4:2]$
Normal Mode	Byte enables	Active during the entire asynchronous access cycle
Select Strobe Mode	Byte enables	Active only during the strobe period of an access cycle

The first mode of operation is normal mode, in which the EM1DQM pins of the EMIF function as byte enables. In this mode, the $\overline{\text{EM1CS}}[4:2]$ pins behaves as typical chip select signals, remaining active for the duration of the asynchronous access. See [Section 24.3.6.1](#) for an example interface with multiple 8-bit devices.

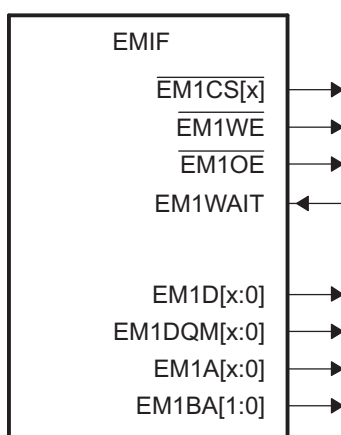
The second mode of operation is select strobe mode, in which the $\overline{\text{EM1CS}}[4:2]$ pins act as a strobe, active only during the strobe period of an access. In this mode, the EM1DQM pins of the EMIF function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in [Table 24-16](#). Refer to [Section 24.3.6.4](#) for the details of asynchronous operations in normal mode, and to [Section 24.3.6.5](#) for the details of asynchronous operations in select strobe mode. The EMIF hardware defaults to normal mode, but can be manually switched to select strobe mode by setting the SS bit in the asynchronous m ($m = 1, 2, 3$, or 4) configuration register (CEnCFG) ($n = 2, 3$, or 4). Throughout the chapter, m can hold the values 1, 2, 3 or 4; and n can hold the values 2, 3, or 4.

The EMIF also provides configurable cycle timing parameters and an extended wait mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

24.3.6.1 Interfacing to Asynchronous Memory

[Figure 24-8](#) shows the EMIF's external pins used in interfacing with an asynchronous device. In $\overline{\text{EM1CS}}[n]$, $n = 2, 3$, or 4 .

Figure 24-8. EMIF Asynchronous Interface

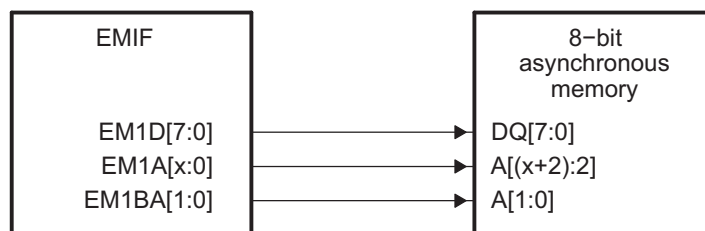


Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pin EM1A[0] always provides the least significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EM1BA[1] and EM1BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Figure 24-9 and Figure 24-10 show the mapping between the EMIF and the connected device's data and address pins for various programmed data bus widths. The data bus width may be configured in the asynchronous *n* configuration register (ASYNC_CS_n_CR).

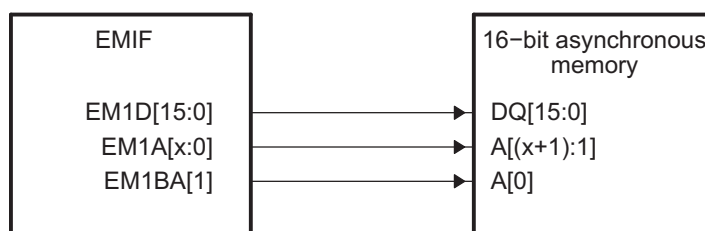
Figure 24-10 shows a common interface between the EMIF and external asynchronous memory.

Figure 24-10 shows an interface between the EMIF and an external memory with byte enables. The EMIF should be operated in either normal mode or select strobe mode when using this interface, so that the EM1DQM signals operate as byte enables.

Figure 24-9. EMIF to 8-bit/16-bit Memory Interface

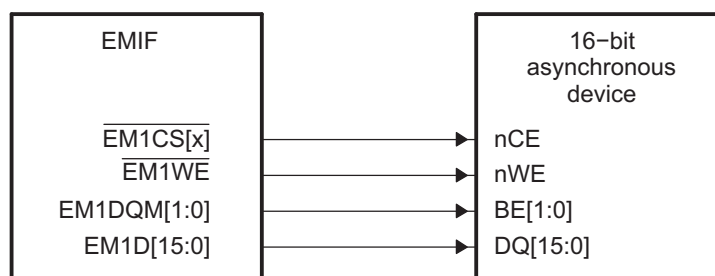


a) EMIF to 8-bit memory interface



b) EMIF to 16-bit memory interface

Figure 24-10. Common Asynchronous Interface



24.3.6.2 Accessing Larger Asynchronous Memories

If a device such as a large asynchronous flash needs to be attached to the EMIF, then GPIO pins may be used to control the flash device's upper address lines.

24.3.6.3 Configuring EMIF for Asynchronous Accesses

The operation of the EMIF's asynchronous interface can be configured by programming the appropriate register fields. The reset value and bit position for each register field can be found in [Section 24.5](#). The following tables list the register fields that can be programmed and describe the purpose of each field. These registers can be programmed prior to accessing the external memory, and the transfer following a write to these registers will use the new configuration.

Table 24-17. Description of the Asynchronous *m* Configuration Register (ASYNC_CS*n*_CR)

Parameter	Description
SS	Select Strobe mode. This bit selects the EMIF's mode of operation in the following way: <ul style="list-style-type: none"> SS = 0 selects Normal Mode <ul style="list-style-type: none"> EM1DQM pins function as byte enables EM1CS[4:2] active for duration of access SS = 1 selects Select Strobe Mode <ul style="list-style-type: none"> EM1DQM pins function as byte enables EM1CS[4:2] acts as a strobe.
EW	Extended Wait Mode enable. <ul style="list-style-type: none"> EW = 0 disables extended wait mode EW = 1 enables extended wait mode When set to 1, the EMIF enables its extended wait mode in which the strobe width of an access cycle can be extended in response to the assertion of the EM1WAIT pin. The WP <i>n</i> bit in the asynchronous wait cycle configuration register (ASYNC_WCCR) controls to polarity of EM1WAIT pin. See Section 24.3.6.6 for more details on this mode of operation.
W_SETUP/R_SETUP	Read/Write setup widths. These fields define the number of EMIF clock cycles of setup time for the address pins (EM1A), byte enables (EM1DQM), and asynchronous chip enable (EM1CS[4:2]) before the read strobe pin (EM1O3) or write strobe pin (EM1WE) falls, minus one cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EM1D). Refer to the asynchronous device's data sheet to determine the appropriate setting for this field.
W_STROBE/R_STROBE	Read/Write strobe widths. These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin (EM1O3) or write strobe pin (EM1WE <i>n</i>), minus one cycle. If Extended Wait Mode is enabled by setting the EW field in the asynchronous <i>n</i> configuration register (ASYNC_CS <i>n</i> _CR), these fields must be set to a value greater than zero. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field.
W_HOLD/R_HOLD	Read/Write hold widths. These fields define the number of EMIF clock cycles of hold time for the address pins (EM1A and EM1BA), byte enables (EM1DQM), and asynchronous chip enable (EM1CS[4:2]) after the read strobe pin (EM1O3) or write strobe pin (EM1WE) rises, minus one cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EM1D). Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field.
TA	Minimum turnaround time. This field defines the minimum number of EMIF clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that will be inserted between asynchronous accesses and SDRAM accesses. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field.
ASIZE	Asynchronous Device Bus Width. This field determines the data bus width of the asynchronous interface in the following way: <ul style="list-style-type: none"> ASIZE = 0 selects an 8-bit bus ASIZE = 1 selects a 16-bit bus ASIZE = 2 selects a 32-bit bus The configuration of ASIZE determines the function of the EM1A and EM1BA pins as described in Section 24.3.6.1 . This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 24.3.2 . For example, a request for a 32-bit word would require four external access when ASIZE = 0. Refer to the data manual of the external asynchronous device to determine the appropriate setting for this field.

Table 24-18. Description of the Asynchronous Wait Cycle Configuration Register (ASYNC_WCCR)

Parameter	Description
WP _n	EM_WAIT Polarity. <ul style="list-style-type: none"> WP_n = 0 selects active-low polarity WP_n = 1 selects active-high polarity <p>When set to 1, the EMIF will wait if the EM1WAIT pin is high. When cleared to 0, the EMIF will wait if the EM1WAIT pin is low. The EMIF must have the Extended Wait Mode enabled for the EM1WAIT pin to affect the width of the strobe period.</p>
MAX_EXT_WAIT	Maximum Extended Wait Cycles. <p>This field configures the number of EMIF clock cycles the EMIF will wait for the EM1WAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIF clock cycles it will wait is determined by the following formula:</p> $\text{Maximum Extended Wait Cycles} = (\text{MAX_EXT_WAIT} + 1) \times 16$ <p>If the EM1WAIT pin is not deactivated within the time specified by this field, the EMIF resumes the access cycle, registering whatever data is on the bus and proceeding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt, if it has been enabled in the EMIF interrupt mask set register (INT_MASK_SET). Refer to Section 24.3.9.1 for more information about EMIF interrupts.</p>

Table 24-19. Description of EMIF Interrupt Mask Set Register (INT_MSK_SET)

Parameter	Description
WR_MASK_SET	Wait Rise Mask Set. Writing a 1 enables an interrupt to be generated when a rising edge on EM1WAIT occurs.
AT_MASK_SET	Asynchronous Timeout Mask Set. Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

Table 24-20. Description of EMIF Interrupt Mast Clear Register (INT_MSK_CLR)

Parameter	Description
WR_MASK_CLR	Wait Rise Mask Clear. Writing a 1 to this bit disables the interrupt, clearing the WR_MASK_SET bit in EMIF interrupt mask set register (INT_MASK_SET).
AT_MASK_CLR	Asynchronous Timeout Mask Clear. Writing a 1 to this bit prevents an interrupt from being generated when an Asynchronous Timeout occurs.

24.3.6.4 Read and Write Operations in Normal Mode

Normal Mode is the asynchronous interface's default mode of operation. It is selected when the SS bit in the asynchronous *n* configuration register (ASYNC_CS_{*n*}_CR) is cleared to 0. In this mode, the EM1DQM pins operate as byte enables. [Section 24.3.6.4.1](#) and [Section 24.3.6.4.2](#) explain the details of read and write operations while in normal mode.

24.3.6.4.1 Asynchronous Read Operations (Normal Mode)

NOTE: During an entire asynchronous read operation, the $\overline{\text{EM1WE}}$ pin is driven high.

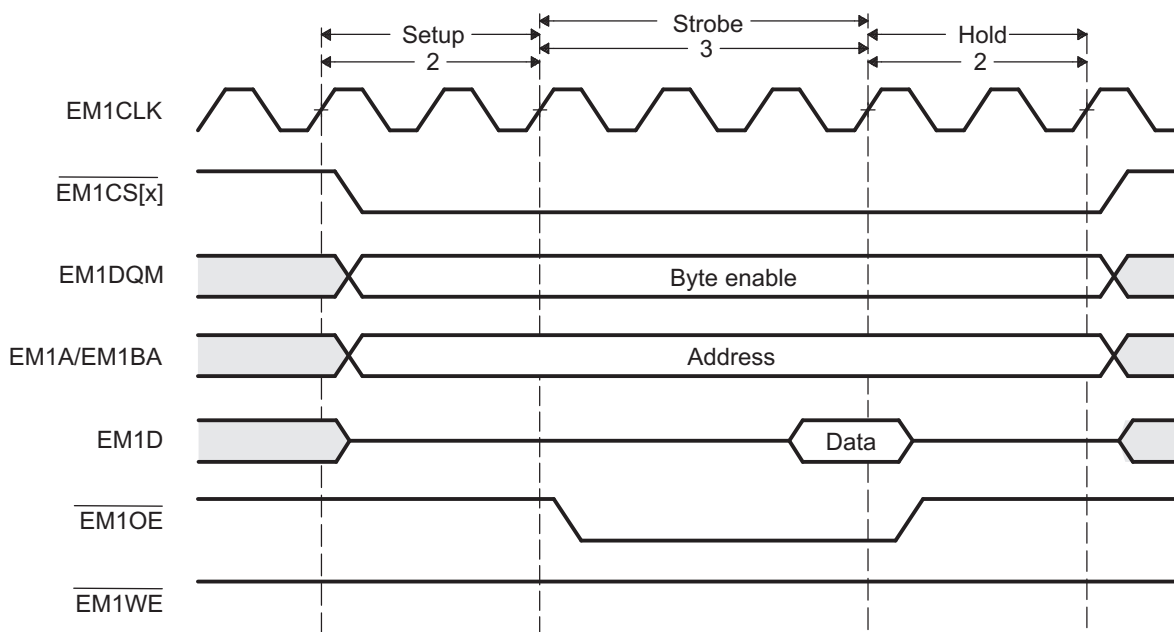
An asynchronous read is performed when any of the requesters mentioned in [Section 24.3.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 24.3.13](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by EMIF until the entire request is fulfilled. The details of an asynchronous read operation in normal mode are described in [Table 24-21](#). Also, [Figure 24-11](#) shows an example timing diagram of a basic read operation.

Table 24-21. Asynchronous Read Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the read operation becomes the highest priority task for EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (ASYNC_CS _{<i>n</i>} _CR). Between each access (write or read) EMIF inserts two cycles of delay even though TA field is programmed as 0. After the EMIF has waited for the turnaround cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: <ul style="list-style-type: none"> The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ASYNC_CS_{<i>n</i>}_CR. The address pins EM1A and EM1BA become valid and carry the values described in Section 24.3.6.1. EMTCS[4:2] falls to enable the external device (if not already low from a previous operation)

Table 24-21. Asynchronous Read Operation in Normal Mode (continued)

Time Interval	Pin Activity in Normal Mode
Strobe period	<p>The following actions occur during the strobe period of a read operation:</p> <ol style="list-style-type: none"> 1. $\overline{\text{EM1OE}}$ falls at the start of the strobe period 2. On the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> • $\overline{\text{EM1OE}}$ rises • The data on the EM1Dx bus is sampled by EMIF. <p>In Figure 24-11, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 24.3.6.6 contains more details on using the EM1WAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> • The address pins EM1A and EM1BA become invalid • $\overline{\text{EM1CS}}[4:2]$ rises (if no more operations are required to complete the current request) <p>The EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

Figure 24-11. Timing Waveform of an Asynchronous Read Cycle in Normal Mode


24.3.6.4.2 Asynchronous Write Operations (Normal Mode)

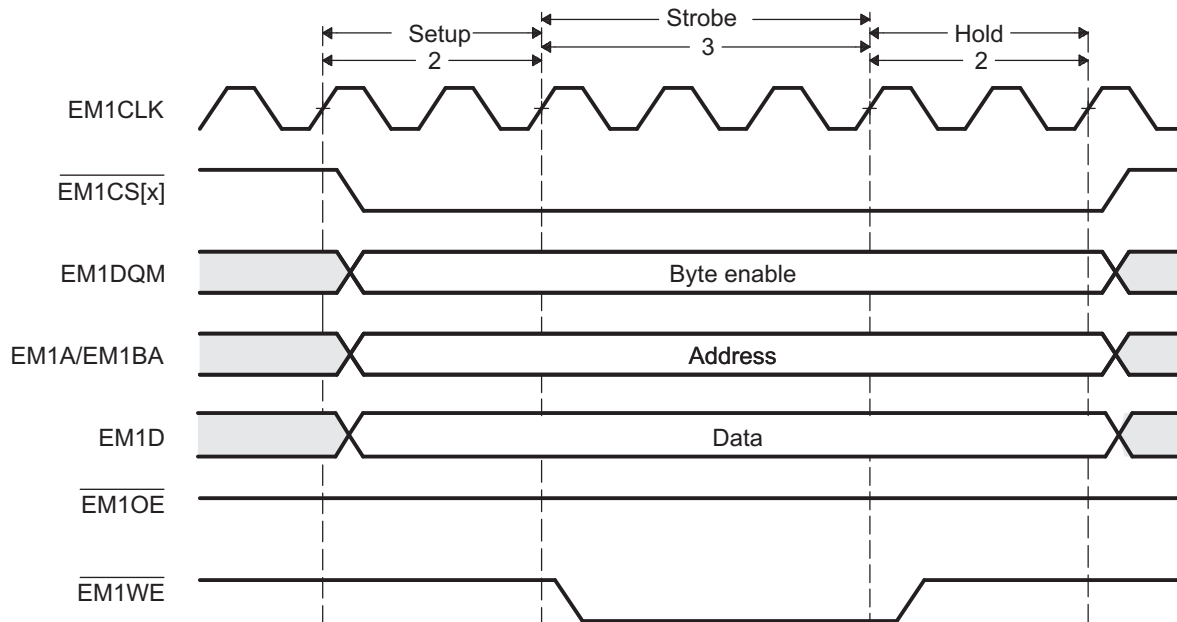
NOTE: During an entire asynchronous write operation, the $\overline{\text{EM1OE}}$ pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in [Section 24.3.2](#) request a write to memory in the asynchronous bank of EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 24.3.13](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in normal mode are described in [Table 24-22](#). Also, [Figure 24-12](#) shows an example timing diagram of a basic write operation.

Table 24-22. Asynchronous Write Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turnaround period	<p>Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous n configuration register (ASYNC_CSn_CR). Between each access (write or read) EMIF inserts two cycles of delay even though TA field is programmed as 0.</p> <p>After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, EMIF terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ASYNC_CSn_CR. The address pins EM1A and EM1BA and the data pins EM1Dx become valid. The EM1A and EM1BA pins carry the values described in Section 24.3.6.1. $\overline{\text{EM1CS}}[4:2]$ falls to enable the external device (if not already low from a previous operation).
Strobe period	<p>The following actions occur at the start of the strobe period of a write operation:</p> <ol style="list-style-type: none"> $\overline{\text{EM1WE}}$ falls The EM1DQM pins become valid as byte enables. <p>The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period:</p> <ol style="list-style-type: none"> $\overline{\text{EM1WE}}$ rises The EM1DQM pins deactivate <p>In Figure 24-12, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 24.3.6.6 contains more details on using the EM1WAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> The address pins EM1Ax and EM1BAx become invalid The data pins become invalid $\overline{\text{EM1CS}}[n]$ ($n = 2, 3, \text{ or } 4$) rises (if no more operations are required to complete the current request) <p>The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

Figure 24-12. Timing Waveform of an Asynchronous Write Cycle in Normal Mode



24.3.6.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIF's second mode of operation. It is selected when the SS bit of the asynchronous n configuration register (ASYNC_CS $_n$ _CR) is set to 1. In this mode, the EM1DQM pins operate as byte enables and the EM1CS $[n]$ ($n = 2, 3, \text{ or } 4$) pin is only active during the strobe period of an access cycle. [Section 24.3.6.4.1](#) and [Section 24.3.6.4.2](#) explain the details of read and write operations while in select strobe mode.

24.3.6.5.1 Asynchronous Read Operations (Select Strobe Mode)

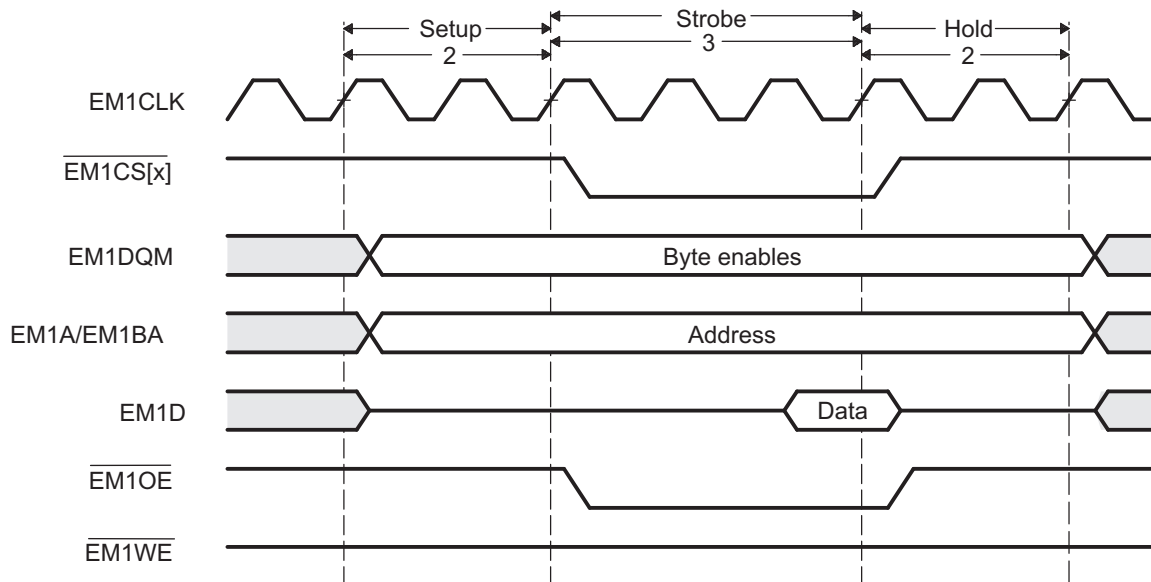
NOTE: During the entirety of an asynchronous read operation, the EM1WEn pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in [Section 24.3.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 24.3.13](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in select strobe mode are described in [Table 24-23](#). Also, [Figure 24-13](#) shows an example timing diagram of a basic read operation.

Table 24-23. Asynchronous Read Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the read operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turnaround cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous n configuration register (ASYNC_CS $_n$ _CR). Between each access (Write or Read) EMIF inserts two cycles of delay even though TA field is programmed as 0. After the EMIF has waited for the turn-around cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: <ul style="list-style-type: none"> The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in ASYNC_CS$_n$_CR. The address pins EM1A and EM1BA become valid and carry the values described in Section 24.3.6.1. The EM1DQM pins become valid as byte enables.
Strobe period	The following actions occur during the strobe period of a read operation: <ol style="list-style-type: none"> EM1CS$[n]$ ($n = 2, 3, \text{ or } 4$) and EM1OE fall at the start of the strobe period On the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> EM1CS$[n]$ ($n = 2, 3, \text{ or } 4$) and EM1OE rise The data on the EM1D bus is sampled by EMIF. <p>In Figure 24-13, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 24.3.6.6 contains more details on using the EM1WAIT pin.</p>
End of the hold period	At the end of the hold period: <ul style="list-style-type: none"> The address pins EM1A and EM1BA become invalid The EM1DQM pins become invalid <p>The EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

Figure 24-13. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode



24.3.6.5.2 Asynchronous Write Operations (Select Strobe Mode)

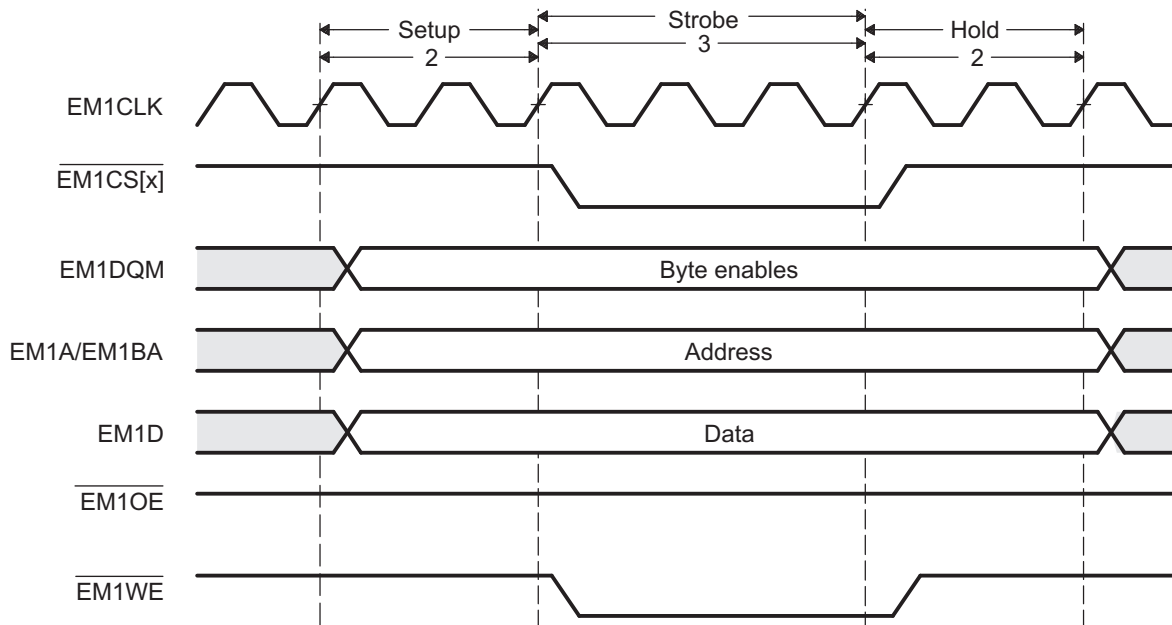
NOTE: During the entirety of an asynchronous write operation, the $\overline{\text{EM1OE}}$ pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in [Section 24.3.2](#) request a write to memory in the asynchronous bank of EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in [Section 24.3.13](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in select strobe mode are described in [Table 24-24](#). Also, [Figure 24-14](#) shows an example timing diagram of a basic write operation.

Table 24-24. Asynchronous Write Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	<p>Once the write operation becomes the highest priority task for the EMIF, the EMIF waits for the programmed number of turnaround cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (ASYNC_CS_n_CR). Between each access (Write or Read) EMIF inserts two cycles of delay even though TA field is programmed as 0.</p> <p>After the EMIF has waited for the turnaround cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in ASYNC_CS_n_CR. The address pins EM1A and EM1BA and the data pins EM1D become valid. The EM1A and EM1BA pins carry the values described in Section 24.3.6.1. The EM1DQM pins become active as byte enables.
Strobe period	<p>The following actions occur at the start of the strobe period of a write operation:</p> <ul style="list-style-type: none"> $\overline{\text{EM1CS}}[n]$ (<i>n</i> = 2, 3, or 4) and $\overline{\text{EM1WE}}$ fall <p>The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period:</p> <ul style="list-style-type: none"> $\overline{\text{EM1CS}}[n]$ (<i>n</i> = 2, 3, or 4) and $\overline{\text{EM1WE}}$ rise <p>In Figure 24-14, EM1WAIT is inactive. If EM1WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 24.3.6.6 contains more details on using the EM1WAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> The address pins EM1A and EM1BA become invalid The data pins become invalid The EM1DQM pins become invalid <p>The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.</p>

Figure 24-14. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode



24.3.6.6 Extended Wait Mode and the EM1WAIT Pin

The EMIF supports the extend wait mode. This is a mode in which the external asynchronous device may assert control over the length of the strobe period. The extended wait mode can be entered by setting the EW bit in the asynchronous n configuration register (ASYNC_CS n _CR ($n = 2, 3$, or 4)). When this bit is set, the EMIF monitors the EM1WAIT pin to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

When the EMIF detects that the EM1WAIT pin has been asserted, it will begin inserting extra strobe cycles into the operation until the EM1WAIT pin is deactivated by the external device. The EMIF will then return to the last cycle of the programmed strobe period and the operation will proceed as usual from this point. Please refer to the device data manual for details on the timing requirements of the EM1WAIT signal.

The EM1WAIT pin cannot be used to extend the strobe period indefinitely. The programmable MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC) determines the maximum number of EM1CLK cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIF proceeds to the hold period of the operation regardless of the state of the EM1WAIT pin. The EMIF can also generate an interrupt upon expiration of this counter. See [Section 24.3.9.1](#) for details on enabling this interrupt.

For the EMIF to function properly in the extended wait mode, the WP n bit of AWCC must be programmed to match the polarity of the EM1WAIT pin. In its reset state of 1, the EMIF will insert wait cycles when the EM1WAIT pin is sampled high. When set to 0, the EMIF will insert wait cycles only when EM1WAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

Finally, a restriction is placed on the strobe period timing parameters when operating in extended wait mode. Specifically, the sum of the W_SETUP and W_STROBE fields must be greater than 4, and the sum of the R_SETUP and R_STROBE fields must be greater than four for the EMIF to recognize the EM1WAIT pin has been asserted. The W_SETUP, W_STROBE, R_SETUP, and R_STROBE fields are in ASYNC_CS n _CR.

24.3.7 Data Bus Parking

The EMIF always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIF issues a read command to the external memory does it stop driving the data bus. After the EMIF latches the last read data, it immediately parks the data bus again.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIF is in the self-refresh state. In this situation, the read operation is not followed by the EMIF parking the data bus. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF is in the self-refresh state, in order to prevent floating inputs on the data bus. External pull-ups, such as 10kΩ resistors, should be placed on the 16 EMIF data bus pins (which do not have internal pull-ups) if it is required to perform reads in this situation. The precise resistor value should be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

For information about the self-refresh state, see [Section 24.3.5.7](#).

24.3.8 Reset and Initialization Considerations

The EMIF memory controller has two active-low reset signals, CHIP_RST_n and MOD_G_RST_n. Both these reset signals are driven by the device system reset signal. This device does not offer the flexibility to reset just the EMIF state machine without also resetting the EMIF controller's memory-mapped registers. As soon as the device system reset is released (driven high), the EMIF memory controller immediately begins its initialization sequence. Command and data stored in the EMIF memory controller FIFOs are lost. Refer to [Section 24.3](#) for more information on conditions that can cause a device system reset to be asserted.

When system reset is released, the EMIF automatically begins running the SDRAM initialization sequence described in [Section 24.3.5.4](#). Even though the initialization procedure is automatic, a special procedure, found in [Section 24.3.5.5](#) must still be followed.

24.3.9 Interrupt Support

The EMIF supports a single interrupt to the CPU. [Section 24.3.9.1](#) details the generation and internal masking of EMIF interrupts.

24.3.9.1 Interrupt Events

There are three conditions that may cause the EMIF to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EM1WAIT signal (wait rise interrupt)
- An asynchronous time out
- Usage of unsupported addressing mode (line trap interrupt)

The wait rise interrupt occurs when a rising edge is detected on EM1WAIT signal. This interrupt generation is not affected by the WP_n bit in the asynchronous wait cycle configuration register (ASYNC_WCCR). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to deassert the EM1WAIT pin within the number of cycles defined by the MAX_EXT_WAIT bit in AWCC (this happens only in extended wait mode). The EMIF supports only linear incrementing and cache line wrap addressing modes. If an access request for an unsupported addressing mode is received, the EMIF will set the LT bit in the EMIF interrupt raw register (INTRAW) and treat the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WR_MASK_SET/AT_MASK_SET/LT_MASK_SET) in the EMIF interrupt mask set register (INT_MASK_SET) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIF interrupt mask clear register (INT_MASK_CLR). The bit fields in both the INT_MASK_SET and INT_MASK_CLR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the INT_MASK_SET and INT_MASK_CLR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIF interrupt raw register (INTRAW) and the IF interrupt mask register (INTMSK) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in INTRAW is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WR_MASKED/AT_MASKED/LT_MASKED) in INTMSK is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in INTRAW clears the INTRAW bit as well as the corresponding bit in INTMSK. [Table 24-25](#) contains a brief summary of the interrupt status and control bit fields. See [Section 24.5](#) for complete details on the register fields.

Table 24-25. Interrupt Monitor and Control Bit Fields

Register Name	Bit Name	Description
EMIF interrupt raw register (INTRAW)	WR	This bit is set when an rising edge on the EM1WAIT signal occurs. Writing a 1 clears the WR bit as well as the WR_MASKED bit in INTMSK.
	AT	This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the AT_MASKED bit in INTMSK.
	LT	This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LT_MASKED bit in INTMSK.
EMIF interrupt mask register (INTMSK)	WR_MASKED	This bit is set only when a rising edge on the EM1WAIT signal occurs and the interrupt has been enabled by writing a 1 to the WR_MASK_SET bit in INT_MASK_SET.
	AT_MASKED	This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the AT_MASK_SET bit in INT_MASK_SET.
	LT_MASKED	This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LT_MASK_SET bit in INT_MASK_SET.
EMIF interrupt mask set register (INT_MASK_SET)	WR_MASK_SET	Writing a 1 to this bit enables the wait rise interrupt.
	AT_MASK_SET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
	LT_MASK_SET	Writing a 1 to this bit enables the line trap interrupt.
EMIF interrupt mask clear register (INT_MASK_CLR)	WR_MASK_CLR	Writing a 1 to this bit disables the wait rise interrupt.
	AT_MASK_CLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.
	LT_MASK_CLR	Writing a 1 to this bit disables the line trap interrupt.

24.3.10 DMA Event Support

The EMIF memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly, by masters and the DMA.

24.3.11 EMIF Signal Multiplexing

For details on the EMIF signal multiplexing, see the *GPIO* chapter, I/O Multiplexing Module section, of this technical reference manual.

24.3.12 Memory Map

For information describing the device memory-map, see your device-specific data manual.

24.3.13 Priority and Arbitration

[Section 24.3.2](#) describes the external prioritization and arbitration among requests from different sources within the microcontroller. The result of this external arbitration is that only one request is presented to the EMIF at a time. Once the EMIF completes a request, the external arbiter then provides the EMIF with the next pending request.

Internally, the EMIF undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A device reset.
- A write to any of the three least significant bytes of the SDRAM configuration register (SDRAM_CR).

Either of these events will cause the EMIF to immediately commence its initialization sequence as described in [Section 24.3.5.4](#).

Once the EMIF has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIF's backlog refresh counter is at the Refresh Must urgency level, the EMIF performs multiple SDRAM auto-refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIF performs a read operation.
3. If the EMIF's backlog refresh counter is at the Refresh Need urgency level, the EMIF performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIF performs a write operation.
5. If the EMIF's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIF performs an SDRAM auto refresh cycle.
6. If the value of the SR bit in SDRAM_CR has been set to 1, the EMIF will enter the self-refresh state as described in [Section 24.3.5.7](#).

After taking one of the actions listed above, the EMIF then returns to the top of the priority list to determine its next action.

Because the EMIF does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See [Section 24.3.5.7](#) for details on the operation of the EMIF when in the self-refresh state.

24.3.14 System Considerations

This section describes various system considerations to keep in mind when operating the EMIF.

24.3.14.1 Asynchronous Request Times

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- t_{RAS} (typically 120 μs) - to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{\text{Refresh Rate}} \times 11$ (typically 15.7 $\mu\text{s} \times 11 = 172.7 \mu\text{s}$) - to avoid refresh violations on the SDRAM.

The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes will require four access cycles using an 8-bit data bus and only two access cycle using a 16-bit data bus. The maximum request size that the EMIF can be sent is 16 words, therefore the maximum number of access cycles per memory request is 64 when the EMIF is configured with an 8-bit data bus. The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EM1WAIT input signal up to a programmed maximum limit. It is up to the user to make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.

24.3.15 Power Management

Power dissipation from the EMIF memory controller may be managed by following methods:

- Self-refresh mode
- Power-down mode
- Gating input clocks to the module off

Gating input clocks off to the EMIF memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clock to EMIF can be turned off through the use of the Global Clock Module (GCM). Before gating clocks off, the EMIF memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the VCLK3 clock domain must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping EMIF memory controller clocks.

24.3.15.1 Power Management Using Self-Refresh Mode

The EMIF memory controller can be placed into a self-refresh state in order to place the attached SDRAM devices into self-refresh mode, which consumes less power for most SDRAM devices. In this state, the attached SDRAM device uses an internal clock to perform its own auto refresh cycles. This maintains the validity of the data in the SDRAM without the need for any external commands. Refer to [Section 24.3.5.7](#) for more details on placing the EMIF into the self-refresh state.

24.3.15.2 Power Management Using Power Down Mode

In the power down mode, the EMIF drives EM1SDCKE low to lower the power consumption. EM1SDCKE goes high when there is a need to send refresh (REFR) commands, after which EM1SDCKE is again driven low. EM1SDCKE remains low until any request arrives. Refer to [Section 24.3.5.8](#) for more details on placing the EMIF in power-down mode.

24.3.16 Emulation Considerations

The EMIF remains fully functional during emulation halts in order to allow emulation access to external memory.

24.4 Example Configuration

This section presents an example of interfacing the EMIF1 to both an SDR SDRAM device and an asynchronous flash device.

24.4.1 Hardware Interface

Figure 24-15 shows the hardware interface between the EMIF, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and a SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between EMIF and the SDRAM is straightforward, but the connection between the EMIF and the flash deserves a detailed look.

The address inputs for the flash are provided by three sources. The A[18:0] address inputs are provided by a combination of the EM1A and EM1BA pins according to Section 24.3.6.1, and a set of GPIO pins. The RD/nBY signal from flash is connected to EM1WAIT pin of the EMIF.

Finally, this example configuration connects the EM1WE pin to the nWE input of the flash and operates the EMIF in select strobe mode.

24.4.2 Software Configuration

The following sections describe how to configure the EMIF registers and bit fields to interface the EMIF with the Samsung K4S641632H-TC(L)70 SDRAM and the SHARP LH28F800BJE-PTTL90 8Mb Flash memory.

24.4.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIF to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of $f_{EM1CLK} = 100$ MHz. Procedure A described in Section 24.3.5.5 is followed which assumes that the SDRAM power-up timing constraint were met during the SDRAM auto-initialization sequence after reset.

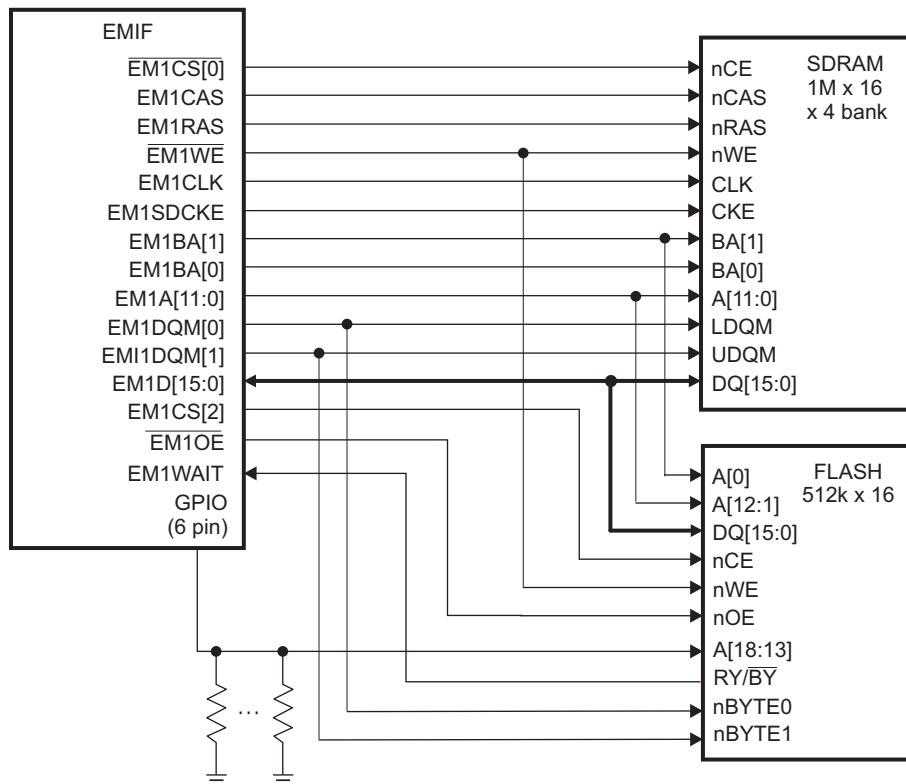
24.4.2.1.1 PLL Programming for EMIF to K4S641632H-TC(L)70 Interface

If the user has programmed the system PLL to provide SYSCLK frequency > 100 MHz, then the user must configure the EMIF1CLKDIV field in the PERSYSCLKDIVSEL register to make EM1CLK = SYSCLK/2 (default configuration). Before doing this, the SDRAM should be placed in self-refresh mode by setting the SR bit in the SDRAM configuration register. Once the EM1CLK frequency has been configured, remove the SDRAM from self-refresh by clearing the SR bit in SDRAM_CR.

Table 24-26. SR Field Value For EMIF to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	1 then 0	To place the EMIF into the self-refresh state

Figure 24-15. Example Configuration Interface



24.4.2.1.2 SDRAM Timing Register (SDRAM_TR) Settings for EMIF to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDRAM_TR) should be programmed first as described in [Table 24-27](#) to satisfy the required timing parameters for the K4S641632H-TC(L)70. Based on these calculations, a value of 6111 4610h should be written to the SDRAM_TR. [Figure 24-16](#) shows a graphical description of how the SDRAM_TR should be programmed.

Table 24-27. SDRAM_TR Field Calculations for EMIF to K4S641632H-TC(L)70 Interface

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_RFC	$T_RFC \geq (t_{RFC} \times f_{EM1CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6
T_RP	$T_RP \geq (t_{RP} \times f_{EM1CLK}) - 1$	$t_{RP} = 20 \text{ ns (min)}$	1
T_RCD	$T_RCD \geq (t_{RCD} \times f_{EM1CLK}) - 1$	$t_{RCD} = 20 \text{ ns (min)}$	1
T_WR	$T_WR \geq (t_{WR} \times f_{EM1CLK}) - 1$	$t_{RDL} = 2 \text{ CLK} = 20 \text{ ns (min)}^{(2)}$	1
T_RAS	$T_RAS \geq (t_{RAS} \times f_{EM1CLK}) - 1$	$t_{RAS} = 49 \text{ ns (min)}$	4
T_RC	$T_RC \geq (t_{RC} \times f_{EM1CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}$	6
T_RRD	$T_RRD \geq (t_{RRD} \times f_{EM1CLK}) - 1$	$t_{RRD} = 14 \text{ ns (min)}$	1

⁽¹⁾ The Samsung datasheet does not specify a t_{RFC} value. Instead, Samsung specifies t_{RC} as the minimum auto refresh period.

⁽²⁾ The Samsung datasheet does not specify a t_{WR} value. Instead, Samsung specifies t_{RDL} as last data in to row precharge minimum delay.

Figure 24-16. SDRAM Timing Register (SDRAM_TR)

31	30	29	28	27	26	24	23	22	21	20	19	18	17	16	
0 0110					001		0	001			0	001			
T_RFC					T_RP		Rsvd		T_RCD			Rsvd		T_WR	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0100				0110			0	001			0000				
T_RAS				T_RC			Rsvd		T_RRD			Reserved			

24.4.2.1.3 SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG) Settings for EMIF to K4S641632H-TC(L)70 Interface

The SDRAM self-refresh exit timing register (SDSRETR) should be programmed second to satisfy the t_{XSR} timing requirement from the K4S641632H-TC(L)70 datasheet. Table 24-28 shows the calculation of the proper value to program into the T_XS field of this register. Based on this calculation, a value of 6h should be written to the SDSRETR. Figure 24-17 shows how the SDSRETR should be programmed.

Table 24-28. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_XS	$T_XS \geq (t_{XSR} \times f_{EM1CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6

⁽¹⁾ The Samsung datasheet does not specify a t_{XSR} value. Instead, Samsung specifies t_{RC} as the minimum required time after CKE going high to complete self-refresh exit.

Figure 24-17. SDRAM Self Refresh Exit Timing Register (SDR_EXT_TMNG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0000 0000 0000 0000															
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000 0000 0000											0 0110				
Reserved											T_XS				

24.4.2.1.4 SDRAM Refresh Control Register (SDRAM_RCR) Settings for EMIF to K4S641632H-TC(L)70 Interface

The SDRAM refresh control register (SDRAM_RCR) should next be programmed to satisfy the required refresh rate of the K4S641632H-TC(L)70. Table 24-29 shows the calculation of the proper value to program into the RR field of this register. Based on this calculation, a value of 61Ah should be written to the SDRAM_RCR. Figure 24-18 shows how the SDRAM_RCR should be programmed.

Table 24-29. RR Calculation for EMIF to K4S641632H-TC(L)70 Interface

Field Name	Formula	Values	Value Calculated for Field
RR	$RR \leq f_{EM1CLK} \times t_{Refresh \text{ Period}} / n_{cycles}$	From SDRAM datasheet: $t_{Refresh \text{ Period}} = 64 \text{ ms}$; $n_{cycles} = 4096$ EMIF clock rate: $f_{EM1CLK} = 100 \text{ MHz}$	$RR = 1562 \text{ cycles} = 61Ah \text{ cycles}$

Figure 24-18. SDRAM Refresh Control Register (SDRAM_RCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0 0000 0000 0000															
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000				0 0110 0001 1010 (61Ah)											
Reserved				RR											

24.4.2.1.5 SDRAM Configuration Register (SDRAM_CR) Settings for EMIF to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDRAM_CR) should be programmed as described in [Table 24-26](#) to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h should be written to the SDRAM_CR. [Figure 24-19](#) shows how the SDRAM_CR should be programmed. The EMIF is now ready to perform read and write accesses to the SDRAM.

Table 24-30. SDRAM_CR Field Values For EMIF to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	0	To avoid placing the EMIF into the self-refresh state
NM	1	To configure the EMIF for a 16-bit data bus
CL	011b	To select a CAS latency of 3
BIT11_9LOCK	1	To allow the CL field to be written
IBANK	010b	To select 4 internal SDRAM banks
PAGESIZE	0	To select a page size of 256 words

Figure 24-19. SDRAM Configuration Register (SDRAM_CR)

31	30	29	28	27	26	25	24
0	0	0			0 0000		
SR	Reserved	Reserved			Reserved		
23	22	21	20	19	18	17	16
		00 0000				0	0
		Reserved				Reserved	Reserved
15	14	13	12	11	10	9	8
0	1	0	0		011		1
Reserved	NM	Reserved	Reserved		CL		BIT11_9LOCK
7	6	5	4	3	2	1	0
0		010		0		000	
Reserved		IBANK		Reserved		PAGESIZE	

24.4.2.2 Configuring the Flash Interface

This section describes how to configure the EMIF to interface with the SHARP LH28F800BJE-PTTL90 8Mb Flash memory with a clock frequency of $f_{EM1CLK} = 100$ MHz.

24.4.2.2.1 Asynchronous 1 Configuration Register (ASYNC_CS2_CFG) Settings for EMIF to LH28F800BJE-PTTL90 Interface

The asynchronous 1 configuration register (ASYNC_CS2_CFG) is the only register that is necessary to program for this asynchronous interface. The SS bit should be set to 1 to enable select strobe command and the ASIZE field should be set to 1 to select a 16-bit interface. The other fields in this register control the shaping of the EMIF signals, and the proper values can be determined by referring to the AC Characteristics in the Flash data manual and the device data manual. Based on the following calculations, a value of 8862 25BDh should be written to ASYNC_CS2_CFG. Table 24-31 and Table 24-32 show the pertinent AC Characteristics for reads and writes to the Flash device, and Figure 24-20 and Figure 24-21 show the associated timing waveforms. Finally, Figure 24-22 shows programming the ASYNC_CS2_CFG with the calculated values.

Table 24-31. AC Characteristics for a Read Access

AC Characteristic	Device	Definition	Min	Max	Unit
t_{SU}	EMIF	Setup time, read EM1D before EM1CLK high	6.5		ns
t_H	EMIF	Data hold time, read EM1D after EM1CLK high	1		ns
t_D	EMIF	Output delay time, EM1CLK high to output signal valid		7	ns
t_{ELQV}	Flash	nCE to Output Delay		90	ns
t_{EHQZ}	Flash	nCE High to Output in High Impedance		55	ns

Table 24-32. AC Characteristics for a Write Access

AC Characteristic	Device	Definition	Min	Max	Unit
t_{AVAV}	Flash	Write Cycle Time	90		ns
t_{ELEH}	Flash	nCE Pulse Width Low	50		ns
t_{EHEL}	Flash	nCE Pulse Width High (not shown in Figure 24-21)	30		ns

Figure 24-20. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms

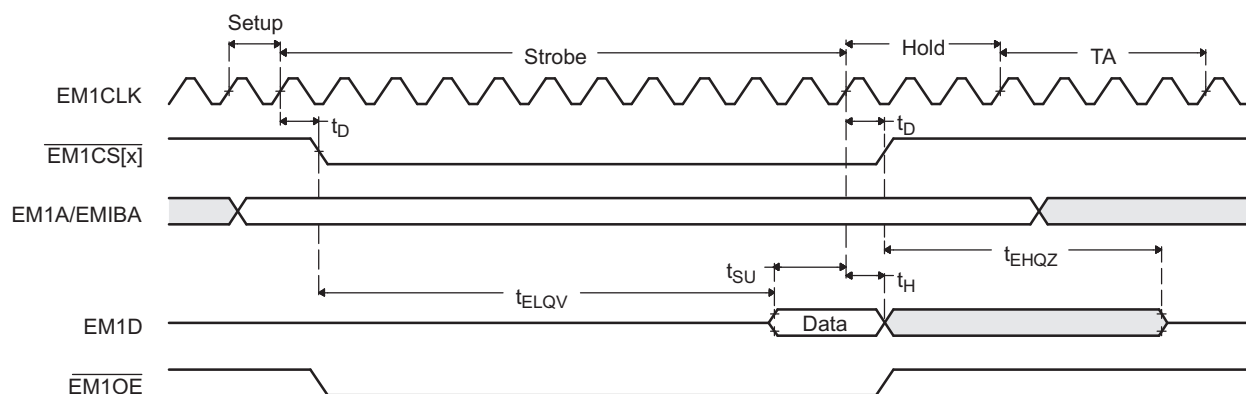
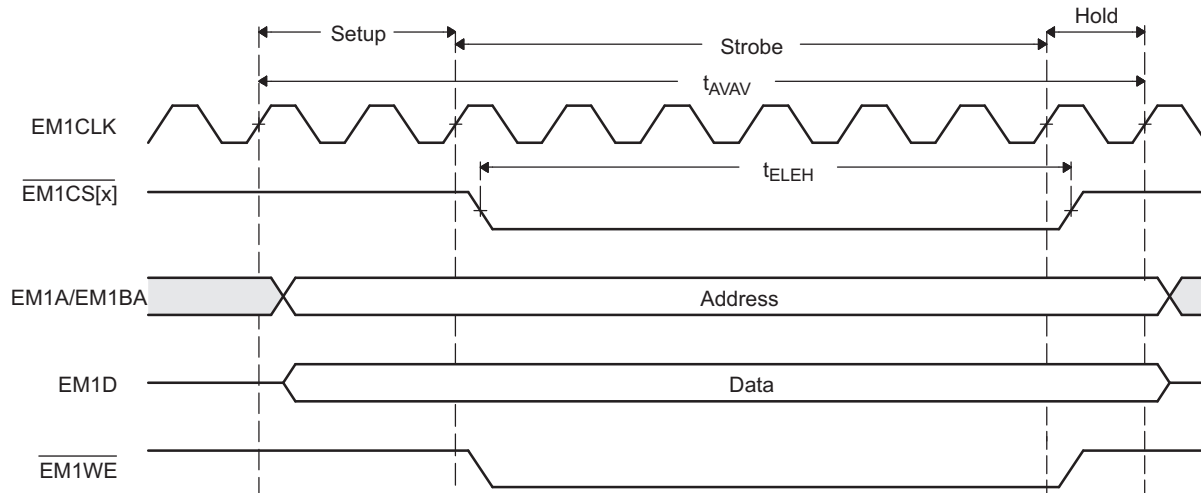


Figure 24-21. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms


The R_STROBE field should be set to meet the following equation:

$$R_STROBE \geq (t_D + t_{ELQV} + t_{SU}) \times f_{EM1CLK} - 1$$

$$R_STROBE \geq (7 \text{ ns} + 90 \text{ ns} + 6.5 \text{ ns}) \times 100 \text{ MHz} - 1$$

$$R_STROBE \geq 9.35$$

$$R_STROBE = 10$$

The R_HOLD field must be large enough to satisfy EMIF Data hold time, t_H :

$$R_HOLD \geq t_H \times f_{EM1CLK} - 1$$

$$R_HOLD \geq 1 \text{ ns} \times 100 \text{ MHz} - 1$$

$$R_HOLD \geq -0.9$$

The R_HOLD field must also combine with the TA field to satisfy the Flash's nCE High to Output in High Impedance time, t_{EHQZ} :

$$R_HOLD + TA \geq (t_D + t_{EHQZ}) \times f_{EM1CLK} - 2$$

$$R_HOLD + TA \geq (7 \text{ ns} + 55 \text{ ns}) \times 100 \text{ MHz} - 2$$

$$R_HOLD + TA \geq 4.2$$

The largest value that can be programmed into the TA field is 3h, therefore the following values can be used:

$$R_HOLD = 2$$

$$TA = 3$$

For Writes, the W_STROBE field should be set to satisfy the Flash's nCE Pulse Width constraint, t_{ELEH} :

$$W_STROBE \geq t_{ELEH} \times f_{EM1CLK} - 1$$

$$W_STROBE \geq 50 \text{ ns} \times 100 \text{ MHz} - 1$$

$$W_STROBE \geq 4$$

The W_SETUP and W_HOLD fields should combine to satisfy the Flash's nCE Pulse Width High constraint, t_{EHEL} :

$$W_SETUP + W_HOLD \geq t_{EHEL} \times f_{EM1CLK} - 2$$

$$W_SETUP + W_HOLD \geq 30 \text{ ns} \times 100 \text{ MHz} - 2$$

$$W_SETUP + W_HOLD \geq 1$$

In addition, the entire Write access length must satisfy the Flash's minimum Write Cycle Time, t_{AVAV} :

$$W_SETUP + W_STROBE + W_HOLD \geq t_{AVAV} \times f_{EM1CLK} - 3$$

$$W_SETUP + W_STROBE + W_HOLD \geq 90 \text{ ns} \times 100 \text{ MHz} - 3$$

$$W_SETUP + W_STROBE + W_HOLD \geq 6$$

Solving the above equations for the Write fields results in the following possible solution:

$$W_SETUP = 1$$

$$W_STROBE = 5$$

$$W_HOLD = 0$$

Adding a 10 ns (1 cycle) margin to each of the periods (excluding TA which is already at its maximum) in this example produces the following recommended values:

$$W_SETUP = 2h$$

$$W_STROBE = 6h$$

$$W_HOLD = 1h$$

$$R_SETUP = 1h$$

$$R_STROBE = 8h$$

$$R_HOLD = 3h$$

$$TA = 3h$$

Figure 24-22. Asynchronous m Configuration Register ($m = 1, 2$) (ASYNC_CS $_n$ _CR($n = 2, 3$))

31		30		29		28		27		26		25		24	
1		0		0010								00			
SS		EW		W_SETUP								W_STROBE			
23		22		21		20		19		18		17		16	
0110								001						0	
W_STROBE								W_HOLD						R_SETUP	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001			001011						011			11		01	
R SETUP			R STROBE						R HOLD			TA		ASIZE	

24.5 Registers

24.5.1 EMIF Base Addresses

Table 24-33. EMIF Base Address Table

Device Registers	Register Name	Start Address	End Address
Emif1Regs	EMIF_REGS	0x0004_7000	0x0004_77FF
Emif2Regs ⁽¹⁾	EMIF_REGS	0x0004_7800	0x0004_7FFF
Emif1ConfigRegs	EMIF1_CONFIG_REGS	0x0005_F480	0x0005_F49F
Emif2ConfigRegs	EMIF2_CONFIG_REGS	0x0005_F4A0	0x0005_F4BF

⁽¹⁾ Only available on CPU1.

24.5.2 EMIF_REGS Registers

Table 24-34 lists the memory-mapped registers for the EMIF_REGS. All register offset addresses not listed in Table 24-34 should be considered as reserved locations and the register contents should not be modified.

Table 24-34. EMIF_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	RCSR	Revision Code and Status Register		Go
2h	ASYNC_WCCR	Async Wait Cycle Config Register		Go
4h	SDRAM_CR	SDRAM (EMxCS0n) Config Register		Go
6h	SDRAM_RCR	SDRAM Refresh Control Register		Go
8h	ASYNC_CS2_CR	Async 1 (EMxCS2n) Config Register		Go
Ah	ASYNC_CS3_CR	Async 2 (EMxCS3n) Config Register		Go
Ch	ASYNC_CS4_CR	Async 3 (EMxCS4n) Config Register		Go
10h	SDRAM_TR	SDRAM Timing Register		Go
18h	TOTAL_SDRAM_AR	Total SDRAM Accesses Register		Go
1Ah	TOTAL_SDRAM_ACTR	Total SDRAM Activate Register		Go
1Eh	SDR_EXT_TMNG	SDRAM SR/PD Exit Timing Register		Go
20h	INT_RAW	Interrupt Raw Register		Go
22h	INT_MSK	Interrupt Masked Register		Go
24h	INT_MSK_SET	Interrupt Mask Set Register		Go
26h	INT_MSK_CLR	Interrupt Mask Clear Register		Go

24.5.2.1 RCSR Register (Offset = 0h) [reset = 40000205h]

RCSR is shown in [Figure 24-23](#) and described in [Table 24-35](#).

Revision Code and Status Register

Figure 24-23. RCSR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-35. RCSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	BE	R	0h	EMIF endian mode. 0: Little Endian. 1: Big Endian.
30	FR	R	1h	EMIF operating rate. 0: Half Rate. 1: Full Rate.
29-16	MODULE_ID	R	0h	EMIF module ID. 0x0000: EMIF_24. 0x000E: EMIF_24 SDRAM. 0x000F: EMIF_24 ASYNC.
15-8	MAJOR_REVISION	R	2h	Major Revision. EMIF code revisions are indicated by a revision code taking the format major_revision.minor_revision.
7-0	MINOR_REVISION	R	5h	Minor Revision. See major_revision field description.

24.5.2.2 ASYNC_WCCR Register (Offset = 2h) [reset = F000080h]

ASYNC_WCCR is shown in [Figure 24-24](#) and described in [Table 24-36](#).

Async Wait Cycle Config Register

Figure 24-24. ASYNC_WCCR Register

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	WP0	RESERVED			
R-0h	R-0h	R-0h	R/W-1h	R-0h			
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
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
MAX_EXT_WAIT							
R/W-80h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-36. ASYNC_WCCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	RESERVED	R	0h	Reserved
29	RESERVED	R	0h	Reserved
28	WP0	R/W	1h	Defines the polarity of the EMxWAIT port.: 0: Wait if EMxWAIT port is low. 1: Wait if EMxWAIT port is high.
27-24	RESERVED	R	0h	Reserved
23-22	RESERVED	R	0h	Reserved
21-20	RESERVED	R	0h	Reserved
19-18	RESERVED	R	0h	Reserved
17-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7-0	MAX_EXT_WAIT	R/W	80h	The EMIF will wait for (max_ext_wait + 1) * 16 clock cycles before an extended asynchronous cycle is terminated.

24.5.2.3 SDRAM_CR Register (Offset = 4h) [reset = 620h]

SDRAM_CR is shown in [Figure 24-25](#) and described in [Table 24-37](#).

SDRAM (EMxCS0n) Config Register

Figure 24-25. SDRAM_CR Register

31	30	29	28	27	26	25	24
SR	PD	PDWR	RESERVED		RESERVED		
R/W-0h	R/W-0h	R/W-0h	R-0h		R-0h		
23	22	21	20	19	18	17	16
RESERVED	RESERVED			RESERVED	RESERVED		RESERVED
R-0h		R-0h		R-0h	R-0h		R-0h
15	14	13	12	11	10	9	8
RESERVED	NM	RESERVED	RESERVED	CL			BIT_11_9_LOCK
R-0h	R/W-0h	R-0h	R-0h	R/W-3h			R=0/W=1-0h
7	6	5	4	3	2	1	0
RESERVED	IBANK			RESERVED	PAGESIZE		
R-0h	R/W-2h			R-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-37. SDRAM_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SR	R/W	0h	Self Refresh. Writing a 1 to this bit will cause connected SDRAM devices to be placed into Self Refresh mode and the EMIF to enter the self refresh state. In this state the EMIF will service all asynchronous memory accesses immediately but any SDRAM access will take at least $t_{\text{ras}} + 1$ cycles due to the time required for the SDRAM devices to out of Self Refresh mode. If an SDRAM access immediately follows the setting of the sr bit, the access will take $t_{\text{ras}} + t_{\text{xs}} + 2$ cycles. If both sr and pd bits are set, the EMIF will go into Self Refresh.
30	PD	R/W	0h	Power Down. Writing a 1 to this bit will cause connected SDRAM devices to be placed into Power Down mode and the EMIF to enter the power down state. If both sr and pd bits are set, the EMIF will go into Self Refresh.
29	PDWR	R/W	0h	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIF to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set.
28-26	RESERVED	R	0h	Reserved
25-23	RESERVED	R	0h	Reserved
22-20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
18-17	RESERVED	R	0h	Reserved
16	RESERVED	R	0h	Reserved
15	RESERVED	R	0h	Reserved
14	NM	R/W	0h	Narrow mode. Set to 1 when system bus width to memory bus width is 2:1 for SDR SDRAM. Set to 0 when system bus width to memory bus width is 1:1 for SDR SDRAM. A write to this field will cause the EMIF to start the SDRAM initialization sequence.
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved

Table 24-37. SDRAM_CR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11-9	CL	R/W	3h	The value of this field defines the CAS latency to be used when accessing connected SDRAM devices. Only CAS latencies of 2 (cl = 2) and 3 (cl = 3) are supported. A write to this field will cause the EMIF to start the SDRAM initialization sequence.
8	BIT_11_9_LOCK	R=0/W=1	0h	Bits 11 to 9 can only be written if this bit is set to 1.
7	RESERVED	R	0h	Reserved
6-4	IBANK	R/W	2h	Defines number of banks inside connected SDRAM devices: 000: 1 bank SDRAM devices. 001: 2 bank SDRAM devices. 010: 3 bank SDRAM devices. 011: Reserved. 1xx: Reserved. A write to this field will cause the EMIF to start the SDRAM initialization sequence.
3	RESERVED	R	0h	Reserved
2-0	PAGESIZE	R/W	0h	Defines the internal page size of connected SDRAM devices: 000: 256-word pages requiring 8 column address bits. 001: 512-word pages requiring 9 column address bits. 010: 1024-word pages requiring 10 column address bits. 011: 2048-word pages requiring 11 column address bits. 1xx: Reserved. A write to this field will cause the EMIF to start the SDRAM initialization sequence.

24.5.2.4 SDRAM_RCR Register (Offset = 6h) [reset = 80h]

SDRAM_RCR is shown in [Figure 24-26](#) and described in [Table 24-38](#).

SDRAM Refresh Control Register

Figure 24-26. SDRAM_RCR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED			
R-0h				R-0h			
15	14	13	12	11	10	9	8
RESERVED			REFRESH_RATE				
R-0h			R/W-80h				
7	6	5	4	3	2	1	0
REFRESH_RATE							
R/W-80h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-38. SDRAM_RCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	RESERVED	R	0h	Reserved
15-13	RESERVED	R	0h	Reserved
12-0	REFRESH_RATE	R/W	80h	Value in this field is used to define the rate at which connected SDRAM devices will be refreshed, as follows: SDRAM refresh rate = EMIF rate/refresh_rate where EMIF rate=clk rate when full_rate=1, or EMIF rate=1/2 clk rate when full_rate=0. Writing a value < 0x0020 to this field will cause it to be loaded with (2 * t_rfc) + 1 value from SDRAM Timing register.

24.5.2.5 ASYNC_CS2_CR Register (Offset = 8h) [reset = 3FFFFFFDh]

ASYNC_CS2_CR is shown in [Figure 24-27](#) and described in [Table 24-39](#).

Async 1 (EMxCS2n) Config Register

Figure 24-27. ASYNC_CS2_CR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS	EW	W_SETUP				W_STROBE						W_HOLD			R_SETUP
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh						R/W-7h			R/W-Fh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R_SETUP			R_STROBE					R_HOLD			TA		ASIZE		
R/W-Fh			R/W-3Fh					R/W-7h			R/W-3h		R/W-1h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-39. ASYNC_CS2_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SS	R/W	0h	Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.
30	EW	R/W	0h	Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.
29-26	W_SETUP	R/W	Fh	Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS2n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.
25-20	W_STROBE	R/W	3Fh	Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
19-17	W_HOLD	R/W	7h	Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS2n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.
16-13	R_SETUP	R/W	Fh	Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS2n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.
12-7	R_STROBE	R/W	3Fh	Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
6-4	R_HOLD	R/W	7h	Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS2n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.
3-2	TA	R/W	3h	Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.
1-0	ASIZE	R/W	1h	Asynchronous Memory Size. Defines the width of the asynchronous device's data bus : 00: 8 Bit data bus. 01: 16 Bit data bus. 10: 32 Bit data bus. 11: Reserved.

24.5.2.6 ASYNC_CS3_CR Register (Offset = Ah) [reset = 3FFFFFFDh]

ASYNC_CS3_CR is shown in Figure 24-28 and described in Table 24-40.

Async 2 (EMxCS3n) Config Register

Figure 24-28. ASYNC_CS3_CR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS	EW	W_SETUP				W_STROBE				W_HOLD				R_SETUP	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh				R/W-7h				R/W-Fh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R_SETUP			R_STROBE					R_HOLD			TA		ASIZE		
R/W-Fh			R/W-3Fh					R/W-7h			R/W-3h		R/W-1h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-40. ASYNC_CS3_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SS	R/W	0h	Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.
30	EW	R/W	0h	Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.
29-26	W_SETUP	R/W	Fh	Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS3n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.
25-20	W_STROBE	R/W	3Fh	Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
19-17	W_HOLD	R/W	7h	Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS3n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.
16-13	R_SETUP	R/W	Fh	Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS3n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.
12-7	R_STROBE	R/W	3Fh	Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
6-4	R_HOLD	R/W	7h	Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS3n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.
3-2	TA	R/W	3h	Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.
1-0	ASIZE	R/W	1h	Asynchronous Memory Size. Defines the width of the asynchronous device's data bus : 00: 8 Bit data bus. 01: 16 Bit data bus. 10: 32 Bit data bus. 11: Reserved.

24.5.2.7 ASYNC_CS4_CR Register (Offset = Ch) [reset = 3FFFFFFDh]

ASYNC_CS4_CR is shown in [Figure 24-29](#) and described in [Table 24-41](#).

Async 3 (EMxCS4n) Config Register

Figure 24-29. ASYNC_CS4_CR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS	EW	W_SETUP				W_STROBE				W_HOLD				R_SETUP	
R/W-0h	R/W-0h	R/W-Fh				R/W-3Fh				R/W-7h				R/W-Fh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R_SETUP			R_STROBE					R_HOLD			TA		ASIZE		
R/W-Fh			R/W-3Fh					R/W-7h			R/W-3h		R/W-1h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-41. ASYNC_CS4_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SS	R/W	0h	Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing.
30	EW	R/W	0h	Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT.
29-26	W_SETUP	R/W	Fh	Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS4n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles.
25-20	W_STROBE	R/W	3Fh	Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
19-17	W_HOLD	R/W	7h	Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS4n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles.
16-13	R_SETUP	R/W	Fh	Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBay, EMxDQMy, and EMxCS4n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles.
12-7	R_STROBE	R/W	3Fh	Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1.
6-4	R_HOLD	R/W	7h	Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBay, EMxDQMy, and EMxCS4n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles.
3-2	TA	R/W	3h	Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles.
1-0	ASIZE	R/W	1h	Asynchronous Memory Size. Defines the width of the asynchronous device's data bus : 00: 8 Bit data bus. 01: 16 Bit data bus. 10: 32 Bit data bus. 11: Reserved.

24.5.2.8 SDRAM_TR Register (Offset = 10h) [reset = 19214610h]

SDRAM_TR is shown in [Figure 24-30](#) and described in [Table 24-42](#).

SDRAM Timing Register

Figure 24-30. SDRAM_TR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-42. SDRAM_TR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	T_RFC	R/W	3h	Minimum number of EMxCLK cycles from Refresh or Load Mode to Refresh or Activate, minus one.
26-24	T_RP	R/W	1h	Minimum number of EMxCLK cycles from Precharge to Activate or Refresh, minus one.
23	RESERVED	R	0h	Reserved
22-20	T_RCD	R/W	2h	Minimum number of EMxCLK cycles from Activate to Read or Write, minus one.
19	RESERVED	R	0h	Reserved
18-16	T_WR	R/W	1h	For SDR, this is equal to minimum number of EMxCLK cycles from last Write transfer to Precharge, minus one.
15-12	T_RAS	R/W	4h	Minimum number of EMxCLK cycles from Activate to Precharge, minus one. $t_{ras} \geq t_{rcd}$.
11-8	T_RC	R/W	6h	Minimum number of EMxCLK cycles from Activate to Activate minus one.
7	RESERVED	R	0h	Reserved
6-4	T_RRD	R/W	1h	Minimum number of EMxCLK cycles from Activate to Activate for a different bank, minus one.
3-0	RESERVED	R	0h	Reserved

24.5.2.9 TOTAL_SDRAM_AR Register (Offset = 18h) [reset = 0h]

TOTAL_SDRAM_AR is shown in [Figure 24-31](#) and described in [Table 24-43](#).

Total SDRAM Accesses Register

Figure 24-31. TOTAL_SDRAM_AR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-43. TOTAL_SDRAM_AR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TOTAL_SDRAM_AR	R	0h	Indicates the total number of accesses to SDRAM from a master (CPUx/CPUX.DMA). This counter is incremented by two for a single access crossing page boundaries.

24.5.2.10 TOTAL_SDRAM_ACTR Register (Offset = 1Ah) [reset = 0h]

TOTAL_SDRAM_ACTR is shown in [Figure 24-32](#) and described in [Table 24-44](#).

Total SDRAM Activate Register

Figure 24-32. TOTAL_SDRAM_ACTR Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-44. TOTAL_SDRAM_ACTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TOTAL_SDRAM_ACTR	R	0h	Indicates the total number of SDRAM accesses which require an activate command.

24.5.2.11 SDR_EXT_TMNG Register (Offset = 1Eh) [reset = 7h]

SDR_EXT_TMNG is shown in [Figure 24-33](#) and described in [Table 24-45](#).

SDRAM SR/PD Exit Timing Register

Figure 24-33. SDR_EXT_TMNG Register

31	30	29	28	27	26	25	24	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								T_XS							
R-0h																R-0h								R/W-7h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-45. SDR_EXT_TMNG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-5	RESERVED	R	0h	Reserved
4-0	T_XS	R/W	7h	This is equal to minimum number of EMxCLK cycles from Self Refresh exit to any command, minus one. For SDR SDRAM, this count should satisfy tXSR.

24.5.2.12 INT_RAW Register (Offset = 20h) [reset = 0h]

INT_RAW is shown in [Figure 24-34](#) and described in [Table 24-46](#).

Interrupt Raw Register

Figure 24-34. INT_RAW 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										WR			LT	AT	
R-0h										R/W=1-0h			R/W=1-0h	R/W=1-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-46. INT_RAW Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5-2	WR	R/W=1	0h	Wait Rise. Set to 1 by hardware to indicate rising edge on the corresponding EMxWAIT has been detected. The WPx bits in the Async Wait Cycle Config register has no effect on these bits. Writing a 1 will clear these bits as well as the wr_masked bits in the Interrupt Masked register. Writing a 0 has no effect.
1	LT	R/W=1	0h	Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size. Writing a 1 will clear this bit as well as the lt_masked bit in the Interrupt Masked register. Writing a 0 has no effect.
0	AT	R/W=1	0h	Asynchronous Timeout. Set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMxWAIT signal did not go inactive within the number of cycles defined by the max_ext_wait field in Async Wait Cycle Config register. Writing a 1 will clear this bit as well as the at_masked bit in the Interrupt Masked register. Writing a 0 has no effect.

24.5.2.13 INT_MSK Register (Offset = 22h) [reset = 0h]

INT_MSK is shown in [Figure 24-35](#) and described in [Table 24-47](#).

Interrupt Masked Register

Figure 24-35. INT_MSK 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		WR_MASKED				LT_MASKED	AT_MASKED
R-0h		R/W=1-0h				R/W=1-0h	R/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-47. INT_MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5-2	WR_MASKED	R/W=1	0h	Masked Wait Rise. Set to 1 by hardware to indicate rising edge on the corresponding EMxWAIT has been detected, only if the wr_mask_set bit in the Interrupt Mask Set register is set to 1. The WPx bits in the Async Wait Cycle Config register has no effect on these bits. Writing a 1 will clear these bits as well as the wr bits in the Interrupt Raw register. Writing a 0 has no effect.
1	LT_MASKED	R/W=1	0h	Masked Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size, only if the lt_mask_set bit in the Interrupt Mask Set register is set to 1. Writing a 1 will clear this bit as well as the lt bit in the Interrupt Raw register. Writing a 0 has no effect.
0	AT_MASKED	R/W=1	0h	Masked Asynchronous Timeout. Set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMxWAIT signal did not go inactive within the number of cycles defined by the max_ext_wait field in Async Wait Cycle Config register, only if the at_mask_set bit in the Interrupt Mask Set register is set to 1. Writing a 1 will clear this bit as well as the at bit in the Interrupt Raw register. Writing a 0 has no effect.

24.5.2.14 INT_MSK_SET Register (Offset = 24h) [reset = 0h]

INT_MSK_SET is shown in [Figure 24-36](#) and described in [Table 24-48](#).

Interrupt Mask Set Register

Figure 24-36. INT_MSK_SET 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		WR_MASK_SET				LT_MASK_SET	AT_MASK_SET
R-0h		R/W=1-0h				R/W=1-0h	R/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-48. INT_MSK_SET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5-2	WR_MASK_SET	R/W=1	0h	Mask set for wr_masked bits in the Interrupt Masked Register. Writing a 1 will enable the interrupts, and set these bits as well as the wr_mask_clr bits in the Interrupt Mask Clear register. Writing a 0 has no effect.
1	LT_MASK_SET	R/W=1	0h	Mask set for lt_masked bit in the Interrupt Masked Register. Writing a 1 will enable the interrupt, and set this bit as well as the lt_mask_clr bit in the Interrupt Mask Clear register. Writing a 0 has no effect.
0	AT_MASK_SET	R/W=1	0h	Mask set for at_masked bit in the Interrupt Masked Register. Writing a 1 will enable the interrupt, and set this bit as well as the at_mask_clr bit in the Interrupt Mask Clear register. Writing a 0 has no effect.

24.5.2.15 INT_MSK_CLR Register (Offset = 26h) [reset = 0h]

INT_MSK_CLR is shown in [Figure 24-37](#) and described in [Table 24-49](#).

Interrupt Mask Clear Register

Figure 24-37. INT_MSK_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		WR_MASK_CLR				LT_MASK_CLR	AT_MASK_CLR
R-0h		R/W=1-0h				R/W=1-0h	R/W=1-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-49. INT_MSK_CLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5-2	WR_MASK_CLR	R/W=1	0h	Mask clear for wr_masked bits in the Interrupt Masked Register. Writing a 1 will disable the interrupts, and clear these bits as well as the wr_mask_set bits in the Interrupt Mask Set register. Writing a 0 has no effect.
1	LT_MASK_CLR	R/W=1	0h	Mask clear for lt_masked bit in the Interrupt Masked Register. Writing a 1 will disable the interrupt, and clear this bit as well as the lt_mask_set bit in the Interrupt Mask Set register. Writing a 0 has no effect.
0	AT_MASK_CLR	R/W=1	0h	Mask clear for at_masked bit in the Interrupt Masked Register. Writing a 1 will disable the interrupt, and clear this bit as well as the at_mask_set bit in the Interrupt Mask Set register. Writing a 0 has no effect.

24.5.3 EMIF1_CONFIG_REGS Registers

Table 24-50 lists the memory-mapped registers for the EMIF1_CONFIG_REGS. All register offset addresses not listed in Table 24-50 should be considered as reserved locations and the register contents should not be modified.

Table 24-50. EMIF1_CONFIG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	EMIF1LOCK	EMIF1 Config Lock Register	EALLOW	Go
2h	EMIF1COMMIT	EMIF1 Config Lock Commit Register	EALLOW	Go
4h	EMIF1MSEL	EMIF1 Master Sel Register	EALLOW	Go
8h	EMIF1ACCPROT0	EMIF1 Config Register 0	EALLOW	Go

24.5.3.1 EMIF1LOCK Register (Offset = 0h) [reset = 0h]

EMIF1LOCK is shown in [Figure 24-38](#) and described in [Table 24-51](#).

EMIF1 Config Lock Register

Figure 24-38. EMIF1LOCK 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							LOCK_EMIF1
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-51. EMIF1LOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	LOCK_EMIF1	R/W	0h	Locks the write to access protection and master select fields for EMIF1: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.

24.5.3.2 EMIF1COMMIT Register (Offset = 2h) [reset = 0h]

EMIF1COMMIT is shown in [Figure 24-39](#) and described in [Table 24-52](#).

EMIF1 Config Lock Commit Register

Figure 24-39. EMIF1COMMIT 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							COMMIT_EMIF 1
R-0h							R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-52. EMIF1COMMIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	COMMIT_EMIF1	R/SOnce	0h	Permanently Locks the write to access protection and master select fields for EMIF1: 0: Write to ACCPROT and Mselect fields are allowed based on value of lock field in EMIF1LOCK register. 1: Write to ACCPROT and Mselect fields are permanently blocked.

24.5.3.3 EMIF1MSEL Register (Offset = 4h) [reset = 0h]

EMIF1MSEL is shown in [Figure 24-40](#) and described in [Table 24-53](#).

EMIF1 Master Sel Register

Figure 24-40. EMIF1MSEL 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
KEY							
R=0/W-0h							
7	6	5	4	3	2	1	0
KEY				RESERVED		MSEL_EMIF1	
R=0/W-0h				R-0h		R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-53. EMIF1MSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	KEY	R=0/W	0h	Writing the value 0x93A5CE7 will allow the writing of the EMIF1Mselect bits, else writes are ignored. Reads will return 0.
3-2	RESERVED	R	0h	Reserved
1-0	MSEL_EMIF1	R/W	0h	Master Select for EMIF1: 00: CPU1 is master but not grabbed. CPU2 can grab the master ownership by changing this value to "10". 01: CPU1 is master. 10: CPU2 is master. 11: CPU1 is master but not grabbed. CPU2 can grab the master ownership by changing this value to "10".

24.5.3.4 EMIF1ACCPROT0 Register (Offset = 8h) [reset = 0h]

EMIF1ACCPROT0 is shown in [Figure 24-41](#) and described in [Table 24-54](#).

EMIF1 Config Register 0

Figure 24-41. EMIF1ACCPROT0 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					DMAWRPROT_ _EMIF1	CPUWRPROT_ _EMIF1	FETCHPROT_ _EMIF1
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-54. EMIF1ACCPROT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-3	RESERVED	R	0h	Reserved
2	DMAWRPROT_EMIF1	R/W	0h	DMA WR Protection For EMIF1: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
1	CPUWRPROT_EMIF1	R/W	0h	CPU WR Protection For EMIF1: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_EMIF1	R/W	0h	Fetch Protection For EMIF1: 0: CPU Fetches are allowed. 1: CPU Fetches are blocked.

24.5.4 EMIF2_CONFIG_REGS Registers

Table 24-55 lists the memory-mapped registers for the EMIF2_CONFIG_REGS. All register offset addresses not listed in Table 24-55 should be considered as reserved locations and the register contents should not be modified.

Table 24-55. EMIF2_CONFIG_REGS Registers

Offset	Acronym	Register Name	Write Protection	Section
0h	EMIF2LOCK	EMIF2 Config Lock Register		Go
2h	EMIF2COMMIT	EMIF2 Config Lock Commit Register	EALLOW	Go
8h	EMIF2ACCPROT0	EMIF2 Config Register 0	EALLOW	Go

24.5.4.1 EMIF2LOCK Register (Offset = 0h) [reset = 0h]

EMIF2LOCK is shown in [Figure 24-42](#) and described in [Table 24-56](#).

EMIF2 Config Lock Register

Figure 24-42. EMIF2LOCK 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							LOCK_EMIF2
R-0h							R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-56. EMIF2LOCK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	LOCK_EMIF2	R/W	0h	Locks the write to access protection fields for EMIF2: 0: Write to ACCPROT and Mselect fields are allowed. 1: Write to ACCPROT and Mselect fields are blocked.

24.5.4.2 EMIF2COMMIT Register (Offset = 2h) [reset = 0h]

EMIF2COMMIT is shown in [Figure 24-43](#) and described in [Table 24-57](#).

EMIF2 Config Lock Commit Register

Figure 24-43. EMIF2COMMIT 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							COMMIT_EMIF2
R-0h							R/SOnce-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-57. EMIF2COMMIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	COMMIT_EMIF2	R/SOnce	0h	Permanently Locks the write to access protection fields for EMIF2: 0: Write to ACCPROT fields are allowed based on value of lock field in EMIF2LOCK register. 1: Write to ACCPROT fields are permanently blocked.

24.5.4.3 EMIF2ACCPROT0 Register (Offset = 8h) [reset = 0h]

EMIF2ACCPROT0 is shown in [Figure 24-44](#) and described in [Table 24-58](#).

EMIF2 Config Register 0

Figure 24-44. EMIF2ACCPROT0 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						CPUWRPROT_ EMIF1	FETCHPROT_ EMIF1
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 24-58. EMIF2ACCPROT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_EMIF1	R/W	0h	CPU WR Protection For EMIF2: 0: CPU Writes are allowed. 1: CPU Writes are blocked.
0	FETCHPROT_EMIF1	R/W	0h	Fetch Protection For EMIF2 0: CPU Fetches are allowed. 1: CPU Fetches are blocked.

Revision History

Changes from December 12, 2014 to June 30, 2015 (from C Revision (December 2014) to D Revision)	Page
• Chapter 1: C28x Processor - No changes for this revision -.....	79
• Chapter 2: System Control	82
• Section 2.3.1 : Inserted a note at the end of this section.....	84
• Table 2-2 : Removed PBIST in the NT12.y row; marked as reserved; changed DCAN to CAN in row INT9.y.	91
• Table 2-4 : Marked INT12.4 row, description as Reserved	93
• Section 2.6.2 : Deleted the bullet, "OSCCLK is switched to INTOSC1" and combined that text with the bullet beginning "PLL is forcefully bypassed..."	100
• Figure 2-3 : Removed PLL021SSP from this figure	101
• Section 2.7.6.2 : Revised text in the ordered list 2a and 2f.....	111
• Section 2.9.1 : Replaced the final paragraph in this section.....	114
• Section 2.9.3 : Added the note after Reset mode	115
• Section 2.10 : Added the last paragraph and list to this section.	117
• Section 2.11.1.3 : Changed "one bit for each LSx memory" to "one bit for each GSx memory"	121
• Table 2-16 : Revised the end address for the cpuTimer0Regs, cpuTimer1Regs, and cpuTimer2Regs.....	152
• Chapter 3: ROM Code and Peripheral Booting	519
• Table 3-5 : In both instances of GPIO255, changed to GPIO254	526
• Revised the sentence beginning "the bit timing registers are programmed..." to align with the 100 kbps bit rate and 20 MHz change in the following table.	566
• Table 3-23 : Changed the values for bit clock and bit rate and created a note following the table.	566
• Chapter 4: Direct Memory Access (DMA)	572
• In the sentence beginning "The PERINTSEL in the MODE register," modified the text in the parenthesis.....	576
• Section 4.4 : Removed HRCAP from Peripheral frame 1.....	583
• Provided a link to the referenced section	583
• Provided a reference for more information.....	583
• Section 4.5.2 : Revised the first sentence in this section.	584
• Chapter 5: Control Law Accelerator (CLA) - No changes for this revision -	608
• Chapter 6: Inter-Processor Communication (IPC)	774
• Table 6-1 : Changed 2kB in both instances to 1K x 16	776
• Chapter 7: General-Purpose Input/Output (GPIO)	843
• Table 7-7 : Changed all "=0x11" to "=11" to indicate a binary, not hexadecimal value.....	851
• Table 7-8 : Changed XTGRIPOUT4 to OUTPUTXBAR4; changed EPWMSYNCO to EXTSYNCO	851
• Table 7-10 : Changed the title of the table from 175-pin to 176-pin.....	852
• Table 7-12 : Changed the header row from 1, 2, 3, 4 to 0, 1, 2, 3.	856
• Figure 7-6 : Reversed EXTSYNIN2 and EXTSYNIN1 connection to ePWM and eCAP Syn Chain	857
• Table 7-13 : Changed the title to Input X-BAR Destinations; changed INPUT5 row, major changes to destinations	857
• Table 7-14 : Included the XbarRegs row; changed the end address of InputXbarRegs from _793F to _791F	858
• Chapter 8: Analog Subsystem - No changes for this revision -.....	1178
• Chapter 9: Analog-to-Digital Converter (ADC)	1193
• Section 9.1.4.4 : Deleted the tables; reworded the corresponding sentence.....	1200
• Figure 9-4 : Revised this graphic	1200
• Chapter 10: Buffered Digital to Analog Converter (DAC) - No changes for this revision -	1373
• Chapter 11: Comparator Subsystem (CMPSS)	1384
• Section 11.1 : Revised the first two sentences of the overview introductory paragraph	1385
• Section 11.1.2 : Added X=H or L to CTRIPx.....	1385
• Section 11.2 : Changed the second entry in the table under Voltages from B < A to A < B.	1386
• Section 11.4 : Added the phrase, 'reference for DACH'	1387
• Chapter 12: Sigma Delta Filter Module (SDFM)	1414
• Figure 12-3 : Changed COMPHx to IEHx; changed COMPLx to IELx	1418
• Table 12-5 : Added this table, the Note preceding it, and the two paragraphs that follow.....	1424

• Section 12.5.2 : Changed the equation to reflect order of sinc filter and data rate of sinc filter	1426
• Section 12.6 : Revised the four comparator low section for clarity	1427
• Section 12.6 : Revised the four comparator high section for clarity	1427
• Section 12.6 : Added Options 1 and 2 this section	1427
• Chapter 13: Enhanced Pulse Width Modulator (ePWM)	1466
• Global Change to chapter: Changed all references from system clock to EPWM clock and from SYSCLK to EPWMCLK.....	1466
• Figure 13-1 : Changed INPUTXBAR6 to INPUTXBAR5	1469
• Changed EPWMSYNCO to EPWMxSYNCO in the last sentence.....	1470
• Section 13.3.6.3.1 Changed SYSCLKOUT to EPWMCLK.....	1515
• Figure 13-41 Figure 13-41 : Revised major portions of this figure.....	1521
• Chapter 14: High-Resolution Pulse Width Modulator (HRPWM)	1758
• Changed all occurrences of SYSCLK and SYSCLKOUT in text, figures, and tables, to EPWMCLK	1758
• Table 14-2 : Abbreviated the first note to TBCLK=EPWMCLK ; removed the 50.0 row	1766
• Table 14-3 : Abbreviated the first note to TBCLK=100 MHz, 10 ns	1767
• Section 14.2.4.2 : Under Assumptions for this example, changed SYSCLK, SYSCLKOUT to TBCLK.....	1768
• Section 14.2.4.3 : Added the final bullet to the duty cycle range section beginning "when using DBREDHR.."	1769
• Figure 14-8 and Figure 14-9 : Changed SYSCLKOUT to EPWMCLK	1771
• Section 14.2.4.4 : Under Assumptions, changed System clock, SYSCLK to TBCLK	1773
• Section 14.2.5 : Added a note referring to DBRED and DBFED values.	1775
• Example 14-6 : Changed all instances of F28M35x to F28x7x in the code	1783
• Chapter 15: Enhanced Capture (eCAP) - No changes for this revision -	1785
• Chapter 16: : Enhanced QEP (eQEP) - No changes for this revision -	1823
• Chapter 17: Serial Peripheral Interface (SPI) - No changes for this revision -	1873
• Chapter 18: Serial Communications Interface (SCI)	1917
• Table 18-3 : Changed 37.5 MHz to 200 MHz.....	1927
• Chapter 19: Inter-Integrated Circuit Module (I2C)	1951
• Table 19-3 : Added this new table.....	1957
• Table 19-4 : Added this new table.....	1959
• Section 19.3.9 : Added this new chapter and diagram	1962
• Chapter 20: Multichannel Buffered Serial Port (McBSP) - No changes for this revision -	1989
• Chapter 21: Controller Area Network (CAN)	2101
• Section 21.15 : In the last paragraph, changed the message RAM base address from 0x2000 to 0x1000.....	2128
• Chapter 22: Universal Serial Bus (USB) Controller	2186
• For all instances of VBUS, subscripted BUS. Each now reads _{VBUS}	2186
• Section 22.1 : Removed the sentence beginning with "DMA access to the FIFO..."	2187
• Section 22.2 : Deleted the last bullet point beginning with "Efficient transfers using..." and both sub-bullets.....	2187
• Modified the text in this section for better clarity.	2188
• Figure 22-2 : Added the GPIOx connector to the diodes.....	2188
• Changed 100K to 100KΩ in this section.....	2188
• Changed the minimum system frequency from 20 MHz to 30 MHz	2189
• Section 22.3.2.2 : In the second paragraph, removed the sentence beginning with "The AUTOCL and AUTORQ bits can be used with DMA..."	2194
• Section 22.3.2.3 : In the first paragraph, removed the sentence beginning with "Furthermore, AUTOSSET can be used with the DMA..."	2194
• Section 22.3.3 : Revised the first paragraph in this section.	2196
• Section 22.6.44 : The field descriptions for the EP1-EP3 fields in these registers have been changed. A value of 0 enables double packet buffering, while a value of 1 disables it	2256
• Section 22.6.45 : The field descriptions for the EP1-EP3 fields in these registers have been changed. A value of 0 enables double packet buffering, while a value of 1 disables it	2257
• Chapter 23: Universal Parallel Port (uPP)	2269
• Figure 23-1 : Changed Universal Serial Port to Universal Parallel Port in the uPPDMA WRITE box.....	2270
• Chapter 24: External Memory Interface (EMIF)	2318
• Changed all occurrences of SoC to MCU	2319

• Section 24.1 : Added this second note regarding EMIF2	2319
• Section 24.1.3 : Replaced SoC with MCU here, and in all other occurrences in this chapter.....	2320
• Section 24.3.2 : Changed all occurrences of crossbar switch to master arbitration block.....	2321
• Section 24.3.6.1 : Deleted the sentence beginning "Additionally, when the EMIF interfaces..."	2337
• Section 24.3.6.4.2 : Replaced EM103 with EM10E	2342
• Section 24.4.1 : Replaced 'two' SHARP flash memory with 'a' SHARP flash memory.....	2354
• Figure 24-15 : Deleted [18:13] from the EMIF block	2355

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com