

# ***TMS320F280013x Real-Time Microcontrollers***

*Technical Reference Manual*

---



Literature Number: SPRUIX1B  
OCTOBER 2022 – REVISED APRIL 2024



# Table of Contents



<b>Read This First</b> .....	51
About This Manual.....	51
Notational Conventions.....	51
Glossary.....	51
Related Documentation From Texas Instruments.....	51
Support Resources.....	51
Trademarks.....	52
<b>1 C2000™ Microcontrollers Software Support</b> .....	53
1.1 Introduction.....	54
1.2 C2000Ware Structure.....	54
1.3 Documentation.....	54
1.4 Devices.....	54
1.5 Libraries.....	54
1.6 Code Composer Studio™ Integrated Development Environment (IDE).....	54
1.7 SysConfig and PinMUX Tool.....	55
<b>2 C28x Processor</b> .....	57
2.1 Introduction.....	58
2.2 C28X Related Collateral.....	58
2.3 Features.....	58
2.4 Floating-Point Unit.....	59
2.5 Trigonometric Math Unit (TMU).....	59
<b>3 System Control and Interrupts</b> .....	61
3.1 Introduction.....	62
3.1.1 SYSCTL Related Collateral.....	62
3.1.2 LOCK Protection on System Configuration Registers.....	62
3.1.3 EALLOW Protection.....	62
3.2 Power Management.....	63
3.3 Device Identification and Configuration Registers.....	63
3.4 Resets.....	63
3.4.1 Reset Sources.....	63
3.4.2 External Reset ( $\overline{XRS}$ ).....	64
3.4.3 Simulate External Reset (SIMRESET.XRS).....	64
3.4.4 Power-On Reset (POR).....	64
3.4.5 Brown-Out-Reset (BOR).....	64
3.4.6 Debugger Reset ( $\overline{SYSRS}$ ).....	65
3.4.7 Simulate CPU Reset.....	65
3.4.8 Watchdog Reset ( $\overline{WDRS}$ ).....	65
3.4.9 NMI Watchdog Reset (NMIWDRS).....	65
3.4.10 DCSM Safe Code Copy Reset (SCCRESET).....	65
3.5 Peripheral Interrupts.....	66
3.5.1 Interrupt Concepts.....	66
3.5.2 Interrupt Architecture.....	66
3.5.3 Interrupt Entry Sequence.....	67
3.5.4 Configuring and Using Interrupts.....	68
3.5.5 PIE Channel Mapping.....	70
3.5.6 PIE Interrupt Priority.....	71
3.5.7 System Error.....	72
3.5.8 Vector Tables.....	73
3.6 Exceptions and Non-Maskable Interrupts.....	77

3.6.1 Configuring and Using NMIs.....	77
3.6.2 Emulation Considerations.....	77
3.6.3 NMI Sources.....	77
3.6.4 Illegal Instruction Trap (ITRAP).....	78
3.6.5 ERRORSTS Pin.....	78
3.7 Clocking.....	78
3.7.1 Clock Sources.....	80
3.7.2 Derived Clocks.....	85
3.7.3 Device Clock Domains.....	86
3.7.4 XCLKOUT.....	87
3.7.5 Clock Connectivity.....	87
3.7.6 Clock Source and PLL Setup.....	88
3.7.7 Using an External Crystal or Resonator.....	89
3.7.8 Using an External Oscillator.....	89
3.7.9 Using an External Resistor (ExtR) With Internal Oscillator.....	89
3.7.10 Choosing PLL Settings.....	90
3.7.11 System Clock Setup.....	91
3.7.12 SYS PLL Bypass.....	91
3.7.13 Clock (OSCCLK) Failure Detection.....	92
3.8 32-Bit CPU Timers 0/1/2.....	94
3.9 Watchdog Timer.....	95
3.9.1 Servicing the Watchdog Timer.....	96
3.9.2 Minimum Window Check.....	96
3.9.3 Watchdog Reset or Watchdog Interrupt Mode.....	97
3.9.4 Watchdog Operation in Low Power Modes.....	97
3.9.5 Emulation Considerations.....	97
3.10 Low-Power Modes.....	98
3.10.1 Clock-Gating Low-Power Modes.....	98
3.10.2 IDLE.....	98
3.10.3 STANDBY.....	99
3.10.4 HALT.....	100
3.11 Memory Controller Module.....	101
3.11.1 Functional Description.....	101
3.12 JTAG.....	105
3.12.1 JTAG Noise and TAP_STATUS.....	105
3.13 System Control Register Configuration Restrictions.....	106
3.14 Software.....	107
3.14.1 SYSCTL Examples.....	107
3.14.2 TIMER Examples.....	107
3.14.3 MEMCFG Examples.....	108
3.14.4 INTERRUPT Examples.....	108
3.14.5 LPM Examples.....	111
3.14.6 WATCHDOG Examples.....	113
3.15 System Control Registers.....	114
3.15.1 SYSCTRL Base Address Table.....	114
3.15.2 ACCESS_PROTECTION_REGS Registers.....	115
3.15.3 CLK_CFG_REGS Registers.....	122
3.15.4 CPU_SYS_REGS Registers.....	142
3.15.5 CPUTIMER_REGS Registers.....	185
3.15.6 DEV_CFG_REGS Registers.....	192
3.15.7 MEM_CFG_REGS Registers.....	213
3.15.8 MEMORY_ERROR_REGS Registers.....	233
3.15.9 NMI_INTRUPT_REGS Registers.....	251
3.15.10 PIE_CTRL_REGS Registers.....	267
3.15.11 SYNC_SOC_REGS Registers.....	295
3.15.12 SYS_STATUS_REGS Registers.....	301
3.15.13 TEST_ERROR_REGS Registers.....	310
3.15.14 UID_REGS Registers.....	314
3.15.15 WD_REGS Registers.....	323
3.15.16 XINT_REGS Registers.....	329
3.15.17 Register to Driverlib Function Mapping.....	338

<b>4 ROM Code and Peripheral Booting</b> .....	<b>351</b>
4.1 Introduction.....	352
4.1.1 ROM Related Collateral.....	352
4.2 Device Boot Sequence.....	353
4.3 Device Boot Modes.....	353
4.3.1 Default Boot Modes.....	353
4.3.2 Custom Boot Modes.....	354
4.4 Device Boot Configurations.....	354
4.4.1 Configuring Boot Mode Pins.....	355
4.4.2 Configuring Boot Mode Table Options.....	357
4.4.3 Boot Mode Example Use Cases.....	358
4.5 Device Boot Flow Diagrams.....	359
4.5.1 Boot Flow.....	359
4.5.2 Emulation Boot Flow.....	361
4.5.3 Standalone Boot Flow.....	362
4.6 Device Reset and Exception Handling.....	363
4.6.1 Reset Causes and Handling.....	363
4.6.2 Exceptions and Interrupts Handling.....	363
4.7 Boot ROM Description.....	364
4.7.1 Boot ROM Configuration Registers.....	364
4.7.2 Entry Points.....	365
4.7.3 Wait Points.....	366
4.7.4 Secure Flash Boot.....	367
4.7.5 Memory Maps.....	370
4.7.6 ROM Tables.....	370
4.7.7 Boot Modes and Loaders.....	371
4.7.8 GPIO Assignments.....	385
4.7.9 Secure ROM Function APIs.....	387
4.7.10 Clock Initializations.....	388
4.7.11 Boot Status Information.....	388
4.7.12 ROM Version.....	390
4.8 Application Notes for Using the Bootloaders.....	390
4.8.1 Bootloader Data Stream Structure.....	390
4.8.2 The C2000 Hex Utility.....	392
<b>5 Dual Code Security Module (DCSM)</b> .....	<b>395</b>
5.1 Introduction.....	396
5.1.1 DCSM Related Collateral.....	396
5.2 Functional Description.....	396
5.2.1 CSM Passwords.....	397
5.2.2 Emulation Code Security Logic (ECSL).....	399
5.2.3 CPU Secure Logic.....	399
5.2.4 Execute-Only Protection.....	399
5.2.5 Password Lock.....	399
5.2.6 JTAGLOCK.....	400
5.2.7 Link Pointer and Zone Select.....	400
5.2.8 C Code Example to Get Zone Select Block Addr for Zone1.....	403
5.3 Flash and OTP Erase/Program.....	403
5.4 Secure Copy Code.....	403
5.5 SecureCRC.....	404
5.6 CSM Impact on Other On-Chip Resources.....	405
5.6.1 RAMOPEN.....	406
5.7 Incorporating Code Security in User Applications.....	407
5.7.1 Environments That Require Security Unlocking.....	407
5.7.2 CSM Password Match Flow.....	407
5.7.3 C Code Example to Unsecure C28x Zone1.....	409
5.7.4 C Code Example to Resecure C28x Zone1.....	409
5.7.5 Environments That Require ECSL Unlocking.....	409
5.7.6 ECSL Password Match Flow.....	409
5.7.7 ECSL Disable Considerations for any Zone.....	411
5.7.8 Device Unique ID.....	411
5.8 Software.....	412

5.8.1 DCSM Examples.....	412
5.9 DCSM Registers.....	412
5.9.1 DCSM Base Address Table.....	412
5.9.2 DCSM_Z1_REGS Registers.....	413
5.9.3 DCSM_Z2_REGS Registers.....	461
5.9.4 DCSM_COMMON_REGS Registers.....	499
5.9.5 DCSM_Z1_OTP Registers.....	521
5.9.6 DCSM_Z2_OTP Registers.....	537
5.9.7 DCSM Registers to Driverlib Functions.....	545
<b>6 Flash Module.....</b>	<b>551</b>
6.1 Introduction to Flash and OTP Memory.....	552
6.1.1 FLASH Related Collateral.....	552
6.1.2 Features.....	552
6.1.3 Flash Tools.....	553
6.1.4 Default Flash Configuration.....	553
6.2 Flash Bank, OTP, and Pump.....	553
6.3 Flash Wrapper.....	554
6.4 Flash and OTP Memory Performance.....	554
6.5 Flash Read Interface.....	555
6.5.1 C28x-Flash Read Interface.....	555
6.6 Flash Erase and Program.....	558
6.6.1 Erase.....	558
6.6.2 Program.....	558
6.6.3 Verify.....	558
6.7 Error Correction Code (ECC) Protection.....	558
6.7.1 Single-Bit Data Error.....	560
6.7.2 Uncorrectable Error.....	561
6.7.3 Mechanism to Check the Correctness of ECC Logic.....	561
6.8 Reserved Locations Within Flash and OTP.....	563
6.9 Migrating an Application from RAM to Flash.....	563
6.10 Procedure to Change the Flash Control Registers.....	564
6.11 Software.....	565
6.11.1 FLASH Examples.....	565
6.12 Flash Registers.....	566
6.12.1 FLASH Base Address Table.....	566
6.12.2 FLASH_CTRL_REGS Registers.....	567
6.12.3 FLASH_ECC_REGS Registers.....	571
6.12.4 FLASH Registers to Driverlib Functions.....	573
<b>7 Dual-Clock Comparator (DCC).....</b>	<b>575</b>
7.1 Introduction.....	576
7.1.1 Features.....	576
7.1.2 Block Diagram.....	576
7.2 Module Operation.....	577
7.2.1 Configuring DCC Counters.....	578
7.2.2 Single-Shot Measurement Mode.....	579
7.2.3 Continuous Monitoring Mode.....	580
7.2.4 Error Conditions.....	581
7.3 Interrupts.....	583
7.4 Software.....	584
7.4.1 DCC Examples.....	584
7.5 DCC Registers.....	586
7.5.1 DCC Base Address Table.....	586
7.5.2 DCC_REGS Registers.....	587
7.5.3 DCC Registers to Driverlib Functions.....	598
<b>8 General-Purpose Input/Output (GPIO).....</b>	<b>601</b>
8.1 Introduction.....	602
8.1.1 GPIO Related Collateral.....	604
8.2 Configuration Overview.....	604
8.3 Digital Inputs on ADC Pins (AIOs).....	605
8.4 Digital Inputs and Outputs on ADC Pins (AGPIOs).....	605
8.5 Digital General-Purpose I/O Control.....	606

8.6 Input Qualification.....	607
8.6.1 No Synchronization (Asynchronous Input).....	607
8.6.2 Synchronization to SYSCLKOUT Only.....	607
8.6.3 Qualification Using a Sampling Window.....	608
8.7 GPIO and Peripheral Muxing.....	611
8.7.1 GPIO Muxing.....	611
8.7.2 Peripheral Muxing.....	613
8.8 Internal Pullup Configuration Requirements.....	614
8.9 Software.....	615
8.9.1 GPIO Examples.....	615
8.9.2 LED Examples.....	616
8.10 GPIO Registers.....	616
8.10.1 GPIO Base Address Table.....	616
8.10.2 GPIO_CTRL_REGS Registers.....	617
8.10.3 GPIO_DATA_REGS Registers.....	689
8.10.4 GPIO_DATA_READ_REGS Registers.....	717
8.10.5 GPIO Registers to Driverlib Functions.....	720
<b>9 Crossbar (X-BAR)</b> .....	<b>723</b>
9.1 Input X-BAR.....	724
9.2 ePWM and GPIO Output X-BAR.....	727
9.2.1 ePWM X-BAR.....	727
9.2.2 GPIO Output X-BAR.....	729
9.2.3 X-BAR Flags.....	731
9.3 XBAR Registers.....	732
9.3.1 XBAR Base Address Table.....	732
9.3.2 INPUT_XBAR_REGS Registers.....	733
9.3.3 XBAR_REGS Registers.....	752
9.3.4 EPWM_XBAR_REGS Registers.....	773
9.3.5 OUTPUT_XBAR_REGS Registers.....	866
9.3.6 Register to Driverlib Function Mapping.....	967
<b>10 Analog Subsystem</b> .....	<b>973</b>
10.1 Introduction.....	974
10.1.1 Features.....	974
10.1.2 Block Diagram.....	974
10.2 Optimizing Power-Up Time.....	976
10.3 Digital Inputs on ADC Pins (AIOs).....	977
10.4 Digital Inputs and Outputs on ADC Pins (AGPIOs).....	977
10.5 Analog Pins and Internal Connections.....	978
10.6 Analog Subsystem Registers.....	980
10.6.1 ASBSYS Base Address Table.....	980
10.6.2 ANALOG_SUBSYS_REGS Registers.....	981
<b>11 Analog-to-Digital Converter (ADC)</b> .....	<b>1003</b>
11.1 Introduction.....	1004
11.1.1 ADC Related Collateral.....	1004
11.1.2 Features.....	1005
11.1.3 Block Diagram.....	1006
11.2 ADC Configurability.....	1007
11.2.1 Clock Configuration.....	1007
11.2.2 Resolution.....	1007
11.2.3 Voltage Reference.....	1008
11.2.4 Signal Mode.....	1008
11.2.5 Expected Conversion Results.....	1009
11.2.6 Interpreting Conversion Results.....	1009
11.3 SOC Principle of Operation.....	1010
11.3.1 SOC Configuration.....	1011
11.3.2 Trigger Operation.....	1011
11.3.3 ADC Acquisition (Sample and Hold) Window.....	1012
11.3.4 ADC Input Models.....	1012
11.3.5 Channel Selection.....	1013
11.4 SOC Configuration Examples.....	1014
11.4.1 Single Conversion from ePWM Trigger.....	1014

11.4.2 Oversampled Conversion from ePWM Trigger.....	1014
11.4.3 Multiple Conversions from CPU Timer Trigger.....	1015
11.4.4 Software Triggering of SOCs.....	1016
11.5 ADC Conversion Priority.....	1016
11.6 Burst Mode.....	1019
11.6.1 Burst Mode Example.....	1019
11.6.2 Burst Mode Priority Example.....	1020
11.7 EOC and Interrupt Operation.....	1021
11.7.1 Interrupt Overflow.....	1022
11.7.2 Continue to Interrupt Mode.....	1022
11.7.3 Early Interrupt Configuration Mode.....	1022
11.8 Post-Processing Blocks.....	1023
11.8.1 PPB Offset Correction.....	1024
11.8.2 PPB Error Calculation.....	1024
11.8.3 PPB Limit Detection and Zero-Crossing Detection.....	1024
11.8.4 PPB Sample Delay Capture.....	1026
11.9 Opens/Shorts Detection Circuit (OSDETECT).....	1027
11.9.1 Implementation.....	1028
11.9.2 Detecting an Open Input Pin.....	1028
11.9.3 Detecting a Shorted Input Pin.....	1028
11.10 Power-Up Sequence.....	1029
11.11 ADC Calibration.....	1029
11.11.1 ADC Zero Offset Calibration.....	1029
11.12 ADC Timings.....	1030
11.12.1 ADC Timing Diagrams.....	1030
11.13 Additional Information.....	1033
11.13.1 Ensuring Synchronous Operation.....	1033
11.13.2 Choosing an Acquisition Window Duration.....	1036
11.13.3 Achieving Simultaneous Sampling.....	1038
11.13.4 Result Register Mapping.....	1038
11.13.5 Internal Temperature Sensor.....	1038
11.13.6 Designing an External Reference Circuit.....	1038
11.13.7 ADC-DAC Loopback Testing.....	1039
11.13.8 Internal Test Mode.....	1040
11.13.9 ADC Gain and Offset Calibration.....	1040
11.14 Software.....	1041
11.14.1 ADC Examples.....	1041
11.15 ADC Registers.....	1045
11.15.1 ADC Base Address Table.....	1045
11.15.2 ADC_RESULT_REGS Registers.....	1046
11.15.3 ADC_REGS Registers.....	1067
11.15.4 ADC Registers to Driverlib Functions.....	1180
<b>12 Comparator Subsystem (CMPSS).....</b>	<b>1185</b>
12.1 Introduction.....	1186
12.1.1 CMPSS Related Collateral.....	1186
12.1.2 Features.....	1186
12.1.3 CMPSS Module Variants.....	1187
12.1.4 Block Diagram.....	1187
12.2 Comparator.....	1188
12.3 Reference DAC.....	1189
12.4 Ramp Generator.....	1190
12.4.1 Ramp Generator Overview.....	1190
12.4.2 Ramp Generator Behavior.....	1191
12.4.3 Ramp Generator Behavior at Corner Cases.....	1192
12.5 Digital Filter.....	1193
12.5.1 Filter Initialization Sequence.....	1194
12.6 Using the CMPSS.....	1194
12.6.1 LATCHCLR, EPWMSYNCPER, and EPWMBLANK Signals.....	1194
12.6.2 Synchronizer, Digital Filter, and Latch Delays.....	1194
12.6.3 Calibrating the CMPSS.....	1195
12.6.4 Enabling and Disabling the CMPSS Clock.....	1195

12.7 CMPSS DAC Output.....	1196
12.8 Software.....	1197
12.8.1 CMPSS Examples.....	1197
12.8.2 CMPSS_LITE Examples.....	1197
12.9 CMPSS Registers.....	1198
12.9.1 CMPSS Base Address Table.....	1198
12.9.2 CMPSS_REGS Registers.....	1199
12.9.3 CMPSS_LITE_REGS Registers.....	1224
12.9.4 CMPSS Registers to Driverlib Functions.....	1239
12.9.5 CMPSS_LITE Registers to Driverlib Functions.....	1241
<b>13 Enhanced Pulse Width Modulator (ePWM).....</b>	<b>1243</b>
13.1 Introduction.....	1244
13.1.1 EPWM Related Collateral.....	1245
13.1.2 Submodule Overview.....	1246
13.2 Configuring Device Pins.....	1251
13.3 ePWM Modules Overview.....	1251
13.4 Time-Base (TB) Submodule.....	1253
13.4.1 Purpose of the Time-Base Submodule.....	1253
13.4.2 Controlling and Monitoring the Time-Base Submodule.....	1254
13.4.3 Calculating PWM Period and Frequency.....	1256
13.4.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules.....	1261
13.4.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules.....	1261
13.4.6 Time-Base Counter Modes and Timing Waveforms.....	1261
13.4.7 Global Load.....	1266
13.5 Counter-Compare (CC) Submodule.....	1268
13.5.1 Purpose of the Counter-Compare Submodule.....	1268
13.5.2 Controlling and Monitoring the Counter-Compare Submodule.....	1269
13.5.3 Operational Highlights for the Counter-Compare Submodule.....	1270
13.5.4 Count Mode Timing Waveforms.....	1271
13.6 Action-Qualifier (AQ) Submodule.....	1274
13.6.1 Purpose of the Action-Qualifier Submodule.....	1274
13.6.2 Action-Qualifier Submodule Control and Status Register Definitions.....	1275
13.6.3 Action-Qualifier Event Priority.....	1277
13.6.4 AQCTLA and AQCTLB Shadow Mode Operations.....	1278
13.6.5 Configuration Requirements for Common Waveforms.....	1280
13.7 Dead-Band Generator (DB) Submodule.....	1287
13.7.1 Purpose of the Dead-Band Submodule.....	1287
13.7.2 Dead-band Submodule Additional Operating Modes.....	1288
13.7.3 Operational Highlights for the Dead-Band Submodule.....	1290
13.8 PWM Chopper (PC) Submodule.....	1294
13.8.1 Purpose of the PWM Chopper Submodule.....	1294
13.8.2 Operational Highlights for the PWM Chopper Submodule.....	1294
13.8.3 Waveforms.....	1295
13.9 Trip-Zone (TZ) Submodule.....	1298
13.9.1 Purpose of the Trip-Zone Submodule.....	1298
13.9.2 Operational Highlights for the Trip-Zone Submodule.....	1299
13.9.3 Generating Trip Event Interrupts.....	1301
13.10 Event-Trigger (ET) Submodule.....	1304
13.10.1 Operational Overview of the ePWM Event-Trigger Submodule.....	1305
13.11 Digital Compare (DC) Submodule.....	1309
13.11.1 Purpose of the Digital Compare Submodule.....	1311
13.11.2 Enhanced Trip Action Using CMPSS.....	1311
13.11.3 Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis.....	1311
13.11.4 Operation Highlights of the Digital Compare Submodule.....	1312
13.12 ePWM Crossbar (X-BAR).....	1319
13.13 Applications to Power Topologies.....	1320
13.13.1 Overview of Multiple Modules.....	1320
13.13.2 Key Configuration Capabilities.....	1321
13.13.3 Controlling Multiple Buck Converters With Independent Frequencies.....	1322
13.13.4 Controlling Multiple Buck Converters With Same Frequencies.....	1324
13.13.5 Controlling Multiple Half H-Bridge (HHB) Converters.....	1326

13.13.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM).....	1328
13.13.7 Practical Applications Using Phase Control Between PWM Modules.....	1330
13.13.8 Controlling a 3-Phase Interleaved DC/DC Converter.....	1331
13.13.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter.....	1334
13.13.10 Controlling a Peak Current Mode Controlled Buck Module.....	1336
13.13.11 Controlling H-Bridge LLC Resonant Converter.....	1337
13.14 Register Lock Protection.....	1338
13.15 High-Resolution Pulse Width Modulator (HRPWM).....	1339
13.15.1 Operational Description of HRPWM.....	1341
13.15.2 SFO Library Software - SFO_TI_Build_V8.lib.....	1362
13.16 Software.....	1365
13.16.1 EPWM Examples.....	1365
13.16.2 HRPWM Examples.....	1369
13.17 ePWM Registers.....	1372
13.17.1 EPWM Base Address Table.....	1372
13.17.2 EPWM_REGS Registers.....	1373
13.17.3 Register to Driverlib Function Mapping.....	1503
<b>14 Enhanced Capture (eCAP).....</b>	<b>1515</b>
14.1 Introduction.....	1516
14.1.1 Features.....	1516
14.1.2 ECAP Related Collateral.....	1516
14.2 Description.....	1517
14.3 Configuring Device Pins for the eCAP.....	1517
14.4 Capture and APWM Operating Mode.....	1520
14.5 Capture Mode Description.....	1522
14.5.1 Event Prescaler.....	1523
14.5.2 Edge Polarity Select and Qualifier.....	1523
14.5.3 Continuous/One-Shot Control.....	1524
14.5.4 32-Bit Counter and Phase Control.....	1525
14.5.5 CAP1-CAP4 Registers.....	1525
14.5.6 eCAP Synchronization.....	1525
14.5.7 Interrupt Control.....	1526
14.5.8 Shadow Load and Lockout Control.....	1528
14.5.9 APWM Mode Operation.....	1528
14.6 Application of the eCAP Module.....	1530
14.6.1 Example 1 - Absolute Time-Stamp Operation Rising-Edge Trigger.....	1530
14.6.2 Example 2 - Absolute Time-Stamp Operation Rising- and Falling-Edge Trigger.....	1531
14.6.3 Example 3 - Time Difference (Delta) Operation Rising-Edge Trigger.....	1532
14.6.4 Example 4 - Time Difference (Delta) Operation Rising- and Falling-Edge Trigger.....	1533
14.7 Application of the APWM Mode.....	1534
14.7.1 Example 1 - Simple PWM Generation (Independent Channels).....	1534
14.8 Software.....	1535
14.8.1 ECAP Examples.....	1535
14.9 eCAP Registers.....	1535
14.9.1 ECAP Base Address Table.....	1535
14.9.2 ECAP_REGS Registers.....	1536
14.9.3 ECAP Registers to Driverlib Functions.....	1554
<b>15 Enhanced Quadrature Encoder Pulse (eQEP).....</b>	<b>1557</b>
15.1 Introduction.....	1558
15.1.1 EQEP Related Collateral.....	1560
15.2 Configuring Device Pins.....	1560
15.3 Description.....	1561
15.3.1 EQEP Inputs.....	1561
15.3.2 Functional Description.....	1564
15.3.3 eQEP Memory Map.....	1565
15.4 Quadrature Decoder Unit (QDU).....	1566
15.4.1 Position Counter Input Modes.....	1566
15.4.2 eQEP Input Polarity Selection.....	1569
15.4.3 Position-Compare Sync Output.....	1569
15.5 Position Counter and Control Unit (PCCU).....	1569
15.5.1 Position Counter Operating Modes.....	1569

15.5.2 Position Counter Latch.....	1572
15.5.3 Position Counter Initialization.....	1574
15.5.4 eQEP Position-compare Unit.....	1575
15.6 eQEP Edge Capture Unit.....	1577
15.7 eQEP Watchdog.....	1581
15.8 eQEP Unit Timer Base.....	1581
15.9 QMA Module.....	1582
15.9.1 Modes of Operation.....	1583
15.9.2 Interrupt and Error Generation.....	1584
15.10 eQEP Interrupt Structure.....	1585
15.11 Software.....	1586
15.11.1 EQEP Examples.....	1586
15.12 eQEP Registers.....	1589
15.12.1 EQEP Base Address Table.....	1589
15.12.2 EQEP_REGS Registers.....	1590
15.12.3 EQEP Registers to Driverlib Functions.....	1627
<b>16 Controller Area Network (CAN).....</b>	<b>1631</b>
16.1 Introduction.....	1632
16.1.1 DCAN Related Collateral.....	1632
16.1.2 Features.....	1632
16.1.3 Block Diagram.....	1633
16.2 Functional Description.....	1635
16.2.1 Configuring Device Pins.....	1635
16.2.2 Address/Data Bus Bridge.....	1635
16.3 Operating Modes.....	1637
16.3.1 Initialization.....	1637
16.3.2 CAN Message Transfer (Normal Operation).....	1638
16.3.3 Test Modes.....	1639
16.4 Multiple Clock Source.....	1643
16.5 Interrupt Functionality.....	1644
16.5.1 Message Object Interrupts.....	1644
16.5.2 Status Change Interrupts.....	1644
16.5.3 Error Interrupts.....	1644
16.5.4 Peripheral Interrupt Expansion (PIE) Module Nomenclature for DCAN Interrupts.....	1644
16.5.5 Interrupt Topologies.....	1645
16.6 Parity Check Mechanism.....	1646
16.6.1 Behavior on Parity Error.....	1646
16.7 Debug Mode.....	1647
16.8 Module Initialization.....	1647
16.9 Configuration of Message Objects.....	1648
16.9.1 Configuration of a Transmit Object for Data Frames.....	1648
16.9.2 Configuration of a Transmit Object for Remote Frames.....	1648
16.9.3 Configuration of a Single Receive Object for Data Frames.....	1648
16.9.4 Configuration of a Single Receive Object for Remote Frames.....	1649
16.9.5 Configuration of a FIFO Buffer.....	1649
16.10 Message Handling.....	1649
16.10.1 Message Handler Overview.....	1650
16.10.2 Receive/Transmit Priority.....	1650
16.10.3 Transmission of Messages in Event Driven CAN Communication.....	1650
16.10.4 Updating a Transmit Object.....	1651
16.10.5 Changing a Transmit Object.....	1651
16.10.6 Acceptance Filtering of Received Messages.....	1652
16.10.7 Reception of Data Frames.....	1652
16.10.8 Reception of Remote Frames.....	1652
16.10.9 Reading Received Messages.....	1652
16.10.10 Requesting New Data for a Receive Object.....	1653
16.10.11 Storing Received Messages in FIFO Buffers.....	1653
16.10.12 Reading from a FIFO Buffer.....	1653
16.11 CAN Bit Timing.....	1655
16.11.1 Bit Time and Bit Rate.....	1655
16.11.2 Configuration of the CAN Bit Timing.....	1660

16.12 Message Interface Register Sets.....	1664
16.12.1 Message Interface Register Sets 1 and 2 (IF1 and IF2).....	1664
16.12.2 Message Interface Register Set 3 (IF3).....	1665
16.13 Message RAM.....	1666
16.13.1 Structure of Message Objects.....	1666
16.13.2 Addressing Message Objects in RAM.....	1669
16.13.3 Message RAM Representation in Debug Mode.....	1670
16.14 Software.....	1671
16.14.1 CAN Examples.....	1671
16.15 CAN Registers.....	1673
16.15.1 CAN Base Address Table.....	1673
16.15.2 CAN_REGS Registers.....	1674
16.15.3 CAN Registers to Driverlib Functions.....	1728
<b>17 Inter-Integrated Circuit Module (I2C).....</b>	<b>1733</b>
17.1 Introduction.....	1734
17.1.1 I2C Related Collateral.....	1734
17.1.2 Features.....	1735
17.1.3 Features Not Supported.....	1735
17.1.4 Functional Overview.....	1736
17.1.5 Clock Generation.....	1737
17.1.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH).....	1738
17.2 Configuring Device Pins.....	1739
17.3 I2C Module Operational Details.....	1739
17.3.1 Input and Output Voltage Levels.....	1739
17.3.2 Selecting Pullup Resistors.....	1739
17.3.3 Data Validity.....	1739
17.3.4 Operating Modes.....	1739
17.3.5 I2C Module START and STOP Conditions.....	1743
17.3.6 Non-repeat Mode versus Repeat Mode.....	1744
17.3.7 Serial Data Formats.....	1744
17.3.8 Clock Synchronization.....	1747
17.3.9 Arbitration.....	1748
17.3.10 Digital Loopback Mode.....	1749
17.3.11 NACK Bit Generation.....	1750
17.4 Interrupt Requests Generated by the I2C Module.....	1750
17.4.1 Basic I2C Interrupt Requests.....	1751
17.4.2 I2C FIFO Interrupts.....	1754
17.5 Resetting or Disabling the I2C Module.....	1754
17.6 Software.....	1755
17.6.1 I2C Examples.....	1755
17.7 I2C Registers.....	1757
17.7.1 I2C Base Address Table.....	1757
17.7.2 I2C_REGS Registers.....	1758
17.7.3 I2C Registers to Driverlib Functions.....	1781
<b>18 Serial Communications Interface (SCI).....</b>	<b>1783</b>
18.1 Introduction.....	1784
18.1.1 Features.....	1784
18.1.2 SCI Related Collateral.....	1785
18.1.3 Block Diagram.....	1785
18.2 Architecture.....	1785
18.3 SCI Module Signal Summary.....	1785
18.4 Configuring Device Pins.....	1787
18.5 Multiprocessor and Asynchronous Communication Modes.....	1787
18.6 SCI Programmable Data Format.....	1788
18.7 SCI Multiprocessor Communication.....	1789
18.7.1 Recognizing the Address Byte.....	1789
18.7.2 Controlling the SCI TX and RX Features.....	1789
18.7.3 Receipt Sequence.....	1789
18.8 Idle-Line Multiprocessor Mode.....	1790
18.8.1 Idle-Line Mode Steps.....	1790
18.8.2 Block Start Signal.....	1791

18.8.3 Wake-Up Temporary (WUT) Flag.....	1791
18.8.4 Receiver Operation.....	1791
18.9 Address-Bit Multiprocessor Mode.....	1792
18.9.1 Sending an Address.....	1792
18.10 SCI Communication Format.....	1793
18.10.1 Receiver Signals in Communication Modes.....	1794
18.10.2 Transmitter Signals in Communication Modes.....	1795
18.11 SCI Port Interrupts.....	1796
18.11.1 Break Detect.....	1798
18.12 SCI Baud Rate Calculations.....	1798
18.13 SCI Enhanced Features.....	1799
18.13.1 SCI FIFO Description.....	1799
18.13.2 SCI Auto-Baud.....	1801
18.13.3 Autobaud-Detect Sequence.....	1801
18.14 Software.....	1802
18.14.1 SCI Examples.....	1802
18.15 SCI Registers.....	1804
18.15.1 SCI Base Address Table.....	1804
18.15.2 SCI_REGS Registers.....	1805
18.15.3 SCI Registers to Driverlib Functions.....	1827
<b>19 Serial Peripheral Interface (SPI).....</b>	<b>1831</b>
19.1 Introduction.....	1832
19.1.1 Features.....	1832
19.1.2 SPI Related Collateral.....	1832
19.1.3 Block Diagram.....	1833
19.2 System-Level Integration.....	1834
19.2.1 SPI Module Signals.....	1834
19.2.2 Configuring Device Pins.....	1835
19.2.3 SPI Interrupts.....	1835
19.3 SPI Operation.....	1837
19.3.1 Introduction to Operation.....	1837
19.3.2 Master Mode.....	1838
19.3.3 Slave Mode.....	1839
19.3.4 Data Format.....	1841
19.3.5 Baud Rate Selection.....	1842
19.3.6 SPI Clocking Schemes.....	1843
19.3.7 SPI FIFO Description.....	1844
19.3.8 SPI High-Speed Mode.....	1845
19.3.9 SPI 3-Wire Mode Description.....	1845
19.4 Programming Procedure.....	1847
19.4.1 Initialization Upon Reset.....	1847
19.4.2 Configuring the SPI.....	1847
19.4.3 Configuring the SPI for High-Speed Mode.....	1848
19.4.4 Data Transfer Example.....	1849
19.4.5 SPI 3-Wire Mode Code Examples.....	1850
19.4.6 SPI STEINV Bit in Digital Audio Transfers.....	1852
19.5 Software.....	1853
19.5.1 SPI Examples.....	1853
19.6 SPI Registers.....	1854
19.6.1 SPI Base Address Table.....	1854
19.6.2 SPI_REGS Registers.....	1855
19.6.3 SPI Registers to Driverlib Functions.....	1873
<b>20 Embedded Pattern Generator (EPG).....</b>	<b>1875</b>
20.1 Introduction.....	1876
20.1.1 Features.....	1876
20.1.2 EPG Block Diagram.....	1876
20.1.3 EPG Related Collateral.....	1877
20.2 Clock Generator Modules.....	1878
20.2.1 DCLK (50% duty cycle clock).....	1878
20.2.2 Clock Stop.....	1879
20.3 Signal Generator Module.....	1880

20.4 EPG Peripheral Signal Mux Selection.....	1883
20.5 EPG Example Use Cases.....	1885
20.5.1 EPG Example: Synchronous Clocks with Offset.....	1885
20.5.2 EPG Example: Serial Data Bit Stream (LSB first).....	1886
20.5.3 EPG Example: Serial Data Bit Stream (MSB first).....	1887
20.6 EPG Interrupt.....	1888
20.7 Software.....	1889
20.7.1 EPG Examples.....	1889
20.8 EPG Registers.....	1890
20.8.1 EPG Base Address Table.....	1890
20.8.2 EPG_REGS Registers.....	1891
20.8.3 EPG_MUX_REGS Registers.....	1919
20.8.4 EPG Registers to Driverlib Functions.....	1924
<b>21 Revision History.....</b>	<b>1927</b>

## List of Figures

Figure 3-1. Device Interrupt Architecture.....	66
Figure 3-2. Interrupt Propagation Path.....	67
Figure 3-3. System Error.....	72
Figure 3-4. ERRORSTS Pin Diagram.....	78
Figure 3-5. Clocking System.....	79
Figure 3-6. System PLL.....	80
Figure 3-7. ExtR Example Schematic.....	81
Figure 3-8. ExtR PCB Layout Example.....	82
Figure 3-9. AUXCLKIN.....	83
Figure 3-10. Single-ended 3.3V External Clock.....	84
Figure 3-11. External Crystal.....	84
Figure 3-12. External Resonator.....	85
Figure 3-13. Missing Clock Detection Logic.....	93
Figure 3-14. CPU Timers.....	94
Figure 3-15. CPU Timer Interrupt Signals and Output Signal.....	94
Figure 3-16. Watchdog Timer Module.....	95
Figure 3-17. Memory Architecture.....	101
Figure 3-18. MAVFLG Register.....	116
Figure 3-19. MAVSET Register.....	117
Figure 3-20. MAVCLR Register.....	118
Figure 3-21. MAVINTEN Register.....	119
Figure 3-22. MCPUFVAVADDR Register.....	120
Figure 3-23. MCPUWRAVADDR Register.....	121
Figure 3-24. CLKCFGLOCK1 Register.....	124
Figure 3-25. CLKSRCCTL1 Register.....	126
Figure 3-26. CLKSRCCTL2 Register.....	128
Figure 3-27. CLKSRCCTL3 Register.....	129
Figure 3-28. SYSPLLCTL1 Register.....	130
Figure 3-29. SYSPLLMULT Register.....	131
Figure 3-30. SYSPLLSTS Register.....	132
Figure 3-31. SYSCLKDIVSEL Register.....	133
Figure 3-32. XCLKOUTDIVSEL Register.....	134
Figure 3-33. LOSPCP Register.....	135
Figure 3-34. MCDCCR Register.....	136
Figure 3-35. X1CNT Register.....	138
Figure 3-36. XTALCR Register.....	139
Figure 3-37. XTALCR2 Register.....	140
Figure 3-38. CLKFAILCFG Register.....	141
Figure 3-39. CPUSYSLOCK1 Register.....	144
Figure 3-40. CPUSYSLOCK2 Register.....	147
Figure 3-41. PIEVERRADDR Register.....	149
Figure 3-42. PCLKCR0 Register.....	150
Figure 3-43. PCLKCR2 Register.....	152
Figure 3-44. PCLKCR3 Register.....	154

Figure 3-45. PCLKCR4 Register.....	155
Figure 3-46. PCLKCR7 Register.....	156
Figure 3-47. PCLKCR8 Register.....	157
Figure 3-48. PCLKCR9 Register.....	158
Figure 3-49. PCLKCR10 Register.....	159
Figure 3-50. PCLKCR13 Register.....	160
Figure 3-51. PCLKCR14 Register.....	161
Figure 3-52. PCLKCR21 Register.....	162
Figure 3-53. PCLKCR27 Register.....	163
Figure 3-54. SIMRESET Register.....	164
Figure 3-55. LPMCR Register.....	165
Figure 3-56. GPIOLPMSEL0 Register.....	166
Figure 3-57. GPIOLPMSEL1 Register.....	169
Figure 3-58. TMR2CLKCTL Register.....	172
Figure 3-59. RESCCLR Register.....	173
Figure 3-60. RESC Register.....	175
Figure 3-61. USER_REG1_SYRSn Register.....	177
Figure 3-62. USER_REG2_SYRSn Register.....	178
Figure 3-63. USER_REG1_XRSn Register.....	179
Figure 3-64. USER_REG2_XRSn Register.....	180
Figure 3-65. USER_REG1_PORESETn Register.....	181
Figure 3-66. USER_REG2_PORESETn Register.....	182
Figure 3-67. USER_REG3_PORESETn Register.....	183
Figure 3-68. USER_REG4_PORESETn Register.....	184
Figure 3-69. TIM Register.....	186
Figure 3-70. PRD Register.....	187
Figure 3-71. TCR Register.....	188
Figure 3-72. TPR Register.....	190
Figure 3-73. TPRH Register.....	191
Figure 3-74. PARTIDL Register.....	194
Figure 3-75. PARTIDH Register.....	195
Figure 3-76. REVID Register.....	196
Figure 3-77. TRIMERRSTS Register.....	197
Figure 3-78. SOFTPRES2 Register.....	198
Figure 3-79. SOFTPRES3 Register.....	200
Figure 3-80. SOFTPRES4 Register.....	201
Figure 3-81. SOFTPRES7 Register.....	202
Figure 3-82. SOFTPRES8 Register.....	203
Figure 3-83. SOFTPRES9 Register.....	204
Figure 3-84. SOFTPRES10 Register.....	205
Figure 3-85. SOFTPRES13 Register.....	206
Figure 3-86. SOFTPRES14 Register.....	207
Figure 3-87. SOFTPRES21 Register.....	208
Figure 3-88. SOFTPRES27 Register.....	209
Figure 3-89. SOFTPRES28 Register.....	210
Figure 3-90. TAP_STATUS Register.....	211
Figure 3-91. ECAPTYPE Register.....	212
Figure 3-92. DxLOCK Register.....	215
Figure 3-93. DxCOMMIT Register.....	216
Figure 3-94. DxACCPROT0 Register.....	217
Figure 3-95. DxACCPROT1 Register.....	218
Figure 3-96. DxTEST Register.....	219
Figure 3-97. DxINIT Register.....	220
Figure 3-98. DxINITDONE Register.....	221
Figure 3-99. DxRAMTEST_LOCK Register.....	222
Figure 3-100. LSxLOCK Register.....	223
Figure 3-101. LSxCOMMIT Register.....	224
Figure 3-102. LSxACCPROT0 Register.....	225
Figure 3-103. LSxTEST Register.....	226
Figure 3-104. LSxINIT Register.....	227
Figure 3-105. LSxINITDONE Register.....	228

Figure 3-106. LSxRAMTEST_LOCK Register.....	229
Figure 3-107. ROM_LOCK Register.....	230
Figure 3-108. ROM_TEST Register.....	231
Figure 3-109. ROM_FORCE_ERROR Register.....	232
Figure 3-110. UCERRFLG Register.....	234
Figure 3-111. UCERRSET Register.....	235
Figure 3-112. UCERRCLR Register.....	236
Figure 3-113. UCCPUREADDR Register.....	237
Figure 3-114. FLUCERRSTATUS Register.....	238
Figure 3-115. FLCERRSTATUS Register.....	239
Figure 3-116. CERRFLG Register.....	241
Figure 3-117. CERRSET Register.....	242
Figure 3-118. CERRCLR Register.....	243
Figure 3-119. CCPUREADDR Register.....	244
Figure 3-120. CERRCNT Register.....	245
Figure 3-121. CERRTHRES Register.....	246
Figure 3-122. CEINTFLG Register.....	247
Figure 3-123. CEINTCLR Register.....	248
Figure 3-124. CEINTSET Register.....	249
Figure 3-125. CEINTEN Register.....	250
Figure 3-126. NMICFG Register.....	252
Figure 3-127. NMIFLG Register.....	253
Figure 3-128. NMIFLGCLR Register.....	255
Figure 3-129. NMIFLGFRC Register.....	257
Figure 3-130. NMIWDCNT Register.....	258
Figure 3-131. NMIWDPRD Register.....	259
Figure 3-132. NMISHDFLG Register.....	260
Figure 3-133. ERRORSTS Register.....	262
Figure 3-134. ERRORSTSCLR Register.....	263
Figure 3-135. ERRORSTSFRC Register.....	264
Figure 3-136. ERRORCTL Register.....	265
Figure 3-137. ERRORLOCK Register.....	266
Figure 3-138. PIECTRL Register.....	269
Figure 3-139. PIEACK Register.....	270
Figure 3-140. PIEIER1 Register.....	271
Figure 3-141. PIEIFR1 Register.....	272
Figure 3-142. PIEIER2 Register.....	273
Figure 3-143. PIEIFR2 Register.....	274
Figure 3-144. PIEIER3 Register.....	275
Figure 3-145. PIEIFR3 Register.....	276
Figure 3-146. PIEIER4 Register.....	277
Figure 3-147. PIEIFR4 Register.....	278
Figure 3-148. PIEIER5 Register.....	279
Figure 3-149. PIEIFR5 Register.....	280
Figure 3-150. PIEIER6 Register.....	281
Figure 3-151. PIEIFR6 Register.....	282
Figure 3-152. PIEIER7 Register.....	283
Figure 3-153. PIEIFR7 Register.....	284
Figure 3-154. PIEIER8 Register.....	285
Figure 3-155. PIEIFR8 Register.....	286
Figure 3-156. PIEIER9 Register.....	287
Figure 3-157. PIEIFR9 Register.....	288
Figure 3-158. PIEIER10 Register.....	289
Figure 3-159. PIEIFR10 Register.....	290
Figure 3-160. PIEIER11 Register.....	291
Figure 3-161. PIEIFR11 Register.....	292
Figure 3-162. PIEIER12 Register.....	293
Figure 3-163. PIEIFR12 Register.....	294
Figure 3-164. SYNCSELECT Register.....	296
Figure 3-165. ADCSOCOUTSELECT Register.....	298
Figure 3-166. SYNCSOCLOCK Register.....	300

Figure 3-167. SYS_ERR_INT_FLG Register.....	302
Figure 3-168. SYS_ERR_INT_CLR Register.....	304
Figure 3-169. SYS_ERR_INT_SET Register.....	306
Figure 3-170. SYS_ERR_MASK Register.....	308
Figure 3-171. CPU_RAM_TEST_ERROR_STS Register.....	311
Figure 3-172. CPU_RAM_TEST_ERROR_STS_CLR Register.....	312
Figure 3-173. CPU_RAM_TEST_ERROR_ADDR Register.....	313
Figure 3-174. UID_PSRAND0 Register.....	315
Figure 3-175. UID_PSRAND1 Register.....	316
Figure 3-176. UID_PSRAND2 Register.....	317
Figure 3-177. UID_PSRAND3 Register.....	318
Figure 3-178. UID_PSRAND4 Register.....	319
Figure 3-179. UID_UNIQUE0 Register.....	320
Figure 3-180. UID_UNIQUE1 Register.....	321
Figure 3-181. UID_CHECKSUM Register.....	322
Figure 3-182. SCSR Register.....	324
Figure 3-183. WDCNTR Register.....	325
Figure 3-184. WDKEY Register.....	326
Figure 3-185. WDCR Register.....	327
Figure 3-186. WDWCR Register.....	328
Figure 3-187. XINT1CR Register.....	330
Figure 3-188. XINT2CR Register.....	331
Figure 3-189. XINT3CR Register.....	332
Figure 3-190. XINT4CR Register.....	333
Figure 3-191. XINT5CR Register.....	334
Figure 3-192. XINT1CTR Register.....	335
Figure 3-193. XINT2CTR Register.....	336
Figure 3-194. XINT3CTR Register.....	337
Figure 4-1. Device Boot Flow.....	360
Figure 4-2. Emulation Boot Flow.....	361
Figure 4-3. CPU Standalone Boot Flow.....	362
Figure 4-4. Overview of SCI Bootloader Operation.....	371
Figure 4-5. Overview of SCI Boot Function.....	372
Figure 4-6. Overview of SPI Bootloader Operation.....	373
Figure 4-7. Data Transfer from EEPROM Flow.....	375
Figure 4-8. EEPROM Device at Address 0x50.....	376
Figure 4-9. Overview of I2C Boot Function.....	377
Figure 4-10. Random Read.....	378
Figure 4-11. Sequential Read.....	378
Figure 4-12. Overview of Parallel GPIO Bootloader Operation.....	379
Figure 4-13. Parallel GPIO Bootloader Handshake Protocol.....	380
Figure 4-14. Overview of Parallel GPIO Boot Function.....	380
Figure 4-15. Parallel GPIO Mode - Host Transfer Flow.....	381
Figure 4-16. 8-Bit Parallel GetWord Function.....	382
Figure 4-17. Overview of CAN-A Bootloader Operation.....	383
Figure 5-1. Storage of Zone-Select Bits in OTP.....	401
Figure 5-2. Location of Zone-Select Block Based on Link-Pointer.....	402
Figure 5-3. CSM Password Match Flow (PMF).....	408
Figure 5-4. ECSL Password Match Flow (PMF).....	410
Figure 5-5. Z1_LINKPOINTER Register.....	415
Figure 5-6. Z1_OTPSECLOCK Register.....	416
Figure 5-7. Z1_JLM_ENABLE Register.....	417
Figure 5-8. Z1_LINKPOINTERERR Register.....	418
Figure 5-9. Z1_GPREG1 Register.....	419
Figure 5-10. Z1_GPREG2 Register.....	420
Figure 5-11. Z1_GPREG3 Register.....	421
Figure 5-12. Z1_GPREG4 Register.....	422
Figure 5-13. Z1_CSMKEY0 Register.....	423
Figure 5-14. Z1_CSMKEY1 Register.....	424
Figure 5-15. Z1_CSMKEY2 Register.....	425
Figure 5-16. Z1_CSMKEY3 Register.....	426

Figure 5-17. Z1_CR Register.....	427
Figure 5-18. Z1_GRABSECT1R Register.....	429
Figure 5-19. Z1_GRABSECT2R Register.....	433
Figure 5-20. Z1_GRABSECT3R Register.....	437
Figure 5-21. Z1_GRABRAM1R Register.....	440
Figure 5-22. Z1_EXEONLYSECT1R Register.....	442
Figure 5-23. Z1_EXEONLYSECT2R Register.....	448
Figure 5-24. Z1_EXEONLYRAM1R Register.....	451
Figure 5-25. Z1_JTAGKEY0 Register.....	453
Figure 5-26. Z1_JTAGKEY1 Register.....	454
Figure 5-27. Z1_JTAGKEY2 Register.....	455
Figure 5-28. Z1_JTAGKEY3 Register.....	456
Figure 5-29. Z1_CMACKKEY0 Register.....	457
Figure 5-30. Z1_CMACKKEY1 Register.....	458
Figure 5-31. Z1_CMACKKEY2 Register.....	459
Figure 5-32. Z1_CMACKKEY3 Register.....	460
Figure 5-33. Z2_LINKPOINTER Register.....	462
Figure 5-34. Z2_OTPSELOCK Register.....	463
Figure 5-35. Z2_LINKPOINTERERR Register.....	464
Figure 5-36. Z2_GPREG1 Register.....	465
Figure 5-37. Z2_GPREG2 Register.....	466
Figure 5-38. Z2_GPREG3 Register.....	467
Figure 5-39. Z2_GPREG4 Register.....	468
Figure 5-40. Z2_CSMKEY0 Register.....	469
Figure 5-41. Z2_CSMKEY1 Register.....	470
Figure 5-42. Z2_CSMKEY2 Register.....	471
Figure 5-43. Z2_CSMKEY3 Register.....	472
Figure 5-44. Z2_CR Register.....	473
Figure 5-45. Z2_GRABSECT1R Register.....	475
Figure 5-46. Z2_GRABSECT2R Register.....	479
Figure 5-47. Z2_GRABSECT3R Register.....	483
Figure 5-48. Z2_GRABRAM1R Register.....	486
Figure 5-49. Z2_EXEONLYSECT1R Register.....	488
Figure 5-50. Z2_EXEONLYSECT2R Register.....	494
Figure 5-51. Z2_EXEONLYRAM1R Register.....	497
Figure 5-52. FLSEM Register.....	500
Figure 5-53. SECTSTAT1 Register.....	501
Figure 5-54. SECTSTAT2 Register.....	504
Figure 5-55. SECTSTAT3 Register.....	507
Figure 5-56. RAMSTAT1 Register.....	510
Figure 5-57. SECERRSTAT Register.....	512
Figure 5-58. SECERRCLR Register.....	513
Figure 5-59. SECERRFRC Register.....	514
Figure 5-60. DENYCODE Register.....	515
Figure 5-61. RAMOPENSTAT Register.....	517
Figure 5-62. RAMOPENFRC Register.....	518
Figure 5-63. RAMOPENCLR Register.....	519
Figure 5-64. RAMOPENLOCK Register.....	520
Figure 5-65. Z1OTP_LINKPOINTER1 Register.....	522
Figure 5-66. Z1OTP_LINKPOINTER2 Register.....	523
Figure 5-67. Z1OTP_LINKPOINTER3 Register.....	524
Figure 5-68. Z1OTP_JLM_ENABLE Register.....	525
Figure 5-69. Z1OTP_GPREG1 Register.....	526
Figure 5-70. Z1OTP_GPREG2 Register.....	527
Figure 5-71. Z1OTP_GPREG3 Register.....	528
Figure 5-72. Z1OTP_GPREG4 Register.....	529
Figure 5-73. Z1OTP_PSWDLOCK Register.....	530
Figure 5-74. Z1OTP_JTAGPSWDH0 Register.....	531
Figure 5-75. Z1OTP_JTAGPSWDH1 Register.....	532
Figure 5-76. Z1OTP_CMACKKEY0 Register.....	533
Figure 5-77. Z1OTP_CMACKKEY1 Register.....	534

Figure 5-78. Z1OTP_CMACKKEY2 Register.....	535
Figure 5-79. Z1OTP_CMACKKEY3 Register.....	536
Figure 5-80. Z2OTP_LINKPOINTER1 Register.....	538
Figure 5-81. Z2OTP_LINKPOINTER2 Register.....	539
Figure 5-82. Z2OTP_LINKPOINTER3 Register.....	540
Figure 5-83. Z2OTP_GPREG1 Register.....	541
Figure 5-84. Z2OTP_GPREG2 Register.....	542
Figure 5-85. Z2OTP_GPREG3 Register.....	543
Figure 5-86. Z2OTP_GPREG4 Register.....	544
Figure 5-87. Z2OTP_PSWDLOCK Register.....	545
Figure 6-1. Flash Interface Block Diagram.....	554
Figure 6-2. Flash Prefetch Mode.....	556
Figure 6-3. ECC Logic Inputs and Outputs.....	559
Figure 6-4. Testing ECC Logic.....	562
Figure 6-5. FRDCNTL Register.....	568
Figure 6-6. FLPROT Register.....	569
Figure 6-7. FRD_INTF_CTRL Register.....	570
Figure 6-8. ECC_ENABLE Register.....	572
Figure 6-9. FECC_CTRL Register.....	573
Figure 7-1. DCC Module Overview.....	576
Figure 7-2. DCC Operation.....	577
Figure 7-3. Counter Relationship.....	581
Figure 7-4. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting.....	581
Figure 7-5. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting.....	582
Figure 7-6. Clock1 Not Present - Results in an Error and Stops Counting.....	582
Figure 7-7. Clock0 Not Present - Results in an Error and Stops Counting.....	583
Figure 7-8. DCCGCTRL Register.....	588
Figure 7-9. DCCCNTSEED0 Register.....	589
Figure 7-10. DCCVALIDSEED0 Register.....	590
Figure 7-11. DCCCNTSEED1 Register.....	591
Figure 7-12. DCCSTATUS Register.....	592
Figure 7-13. DCCCNT0 Register.....	593
Figure 7-14. DCCVALID0 Register.....	594
Figure 7-15. DCCCNT1 Register.....	595
Figure 7-16. DCCCLKSRC1 Register.....	596
Figure 7-17. DCCCLKSRC0 Register.....	598
Figure 8-1. GPIO Logic for a Single Pin.....	603
Figure 8-2. Input Qualification Using a Sampling Window.....	608
Figure 8-3. Input Qualifier Clock Cycles.....	610
Figure 8-4. GPACTRL Register.....	619
Figure 8-5. GPAQSEL1 Register.....	620
Figure 8-6. GPAQSEL2 Register.....	622
Figure 8-7. GPAMUX1 Register.....	624
Figure 8-8. GPAMUX2 Register.....	625
Figure 8-9. GPADIR Register.....	626
Figure 8-10. GPAPUD Register.....	628
Figure 8-11. GPAINV Register.....	630
Figure 8-12. GPAODR Register.....	632
Figure 8-13. GPAAMSEL Register.....	634
Figure 8-14. GPAGMUX1 Register.....	636
Figure 8-15. GPAGMUX2 Register.....	637
Figure 8-16. GPALOCK Register.....	638
Figure 8-17. GPACR Register.....	640
Figure 8-18. GPBCTRL Register.....	642
Figure 8-19. GPBQSEL1 Register.....	643
Figure 8-20. GPBMUX1 Register.....	645
Figure 8-21. GPBDIR Register.....	646
Figure 8-22. GPBPUD Register.....	648
Figure 8-23. GPBINV Register.....	650
Figure 8-24. GPBODR Register.....	652
Figure 8-25. GPBGMUX1 Register.....	654

Figure 8-26. GPBLOCK Register.....	655
Figure 8-27. GPBCR Register.....	657
Figure 8-28. GPHCTRL Register.....	659
Figure 8-29. GPHQSEL1 Register.....	660
Figure 8-30. GPHQSEL2 Register.....	662
Figure 8-31. GPHMUX1 Register.....	664
Figure 8-32. GPHMUX2 Register.....	665
Figure 8-33. GPHDIR Register.....	666
Figure 8-34. GPHPUD Register.....	668
Figure 8-35. GPHINV Register.....	672
Figure 8-36. GPHODR Register.....	675
Figure 8-37. GPHAMSEL Register.....	677
Figure 8-38. GPHGMUX1 Register.....	681
Figure 8-39. GPHGMUX2 Register.....	682
Figure 8-40. GPHLOCK Register.....	683
Figure 8-41. GPHCR Register.....	686
Figure 8-42. GPADAT Register.....	690
Figure 8-43. GPASET Register.....	692
Figure 8-44. GPACLEAR Register.....	694
Figure 8-45. GPATOGGLE Register.....	696
Figure 8-46. GPBDAT Register.....	698
Figure 8-47. GPBSET Register.....	700
Figure 8-48. GPBCLEAR Register.....	702
Figure 8-49. GPBTOGGLE Register.....	704
Figure 8-50. GPHDAT Register.....	706
Figure 8-51. GPHSET Register.....	711
Figure 8-52. GPHCLEAR Register.....	713
Figure 8-53. GPHTOGGLE Register.....	715
Figure 8-54. GPADAT_R Register.....	718
Figure 8-55. GPBDAT_R Register.....	719
Figure 8-56. GPHDAT_R Register.....	720
Figure 9-1. Input X-BAR.....	725
Figure 9-2. ePWM X-BAR Architecture - Single Output.....	727
Figure 9-3. GPIO Output X-BAR Architecture.....	729
Figure 9-4. X-BAR Input Sources.....	731
Figure 9-5. INPUT1SELECT Register.....	734
Figure 9-6. INPUT2SELECT Register.....	735
Figure 9-7. INPUT3SELECT Register.....	736
Figure 9-8. INPUT4SELECT Register.....	737
Figure 9-9. INPUT5SELECT Register.....	738
Figure 9-10. INPUT6SELECT Register.....	739
Figure 9-11. INPUT7SELECT Register.....	740
Figure 9-12. INPUT8SELECT Register.....	741
Figure 9-13. INPUT9SELECT Register.....	742
Figure 9-14. INPUT10SELECT Register.....	743
Figure 9-15. INPUT11SELECT Register.....	744
Figure 9-16. INPUT12SELECT Register.....	745
Figure 9-17. INPUT13SELECT Register.....	746
Figure 9-18. INPUT14SELECT Register.....	747
Figure 9-19. INPUT15SELECT Register.....	748
Figure 9-20. INPUT16SELECT Register.....	749
Figure 9-21. INPUTSELECTLOCK Register.....	750
Figure 9-22. XBARFLG1 Register.....	753
Figure 9-23. XBARFLG2 Register.....	756
Figure 9-24. XBARFLG3 Register.....	760
Figure 9-25. XBARFLG4 Register.....	762
Figure 9-26. XBARCLR1 Register.....	764
Figure 9-27. XBARCLR2 Register.....	766
Figure 9-28. XBARCLR3 Register.....	769
Figure 9-29. XBARCLR4 Register.....	771
Figure 9-30. TRIP4MUX0TO15CFG Register.....	775

Figure 9-31. TRIP4MUX16TO31CFG Register.....	778
Figure 9-32. TRIP5MUX0TO15CFG Register.....	781
Figure 9-33. TRIP5MUX16TO31CFG Register.....	784
Figure 9-34. TRIP7MUX0TO15CFG Register.....	787
Figure 9-35. TRIP7MUX16TO31CFG Register.....	790
Figure 9-36. TRIP8MUX0TO15CFG Register.....	793
Figure 9-37. TRIP8MUX16TO31CFG Register.....	796
Figure 9-38. TRIP9MUX0TO15CFG Register.....	799
Figure 9-39. TRIP9MUX16TO31CFG Register.....	802
Figure 9-40. TRIP10MUX0TO15CFG Register.....	805
Figure 9-41. TRIP10MUX16TO31CFG Register.....	808
Figure 9-42. TRIP11MUX0TO15CFG Register.....	811
Figure 9-43. TRIP11MUX16TO31CFG Register.....	814
Figure 9-44. TRIP12MUX0TO15CFG Register.....	817
Figure 9-45. TRIP12MUX16TO31CFG Register.....	820
Figure 9-46. TRIP4MUXENABLE Register.....	823
Figure 9-47. TRIP5MUXENABLE Register.....	828
Figure 9-48. TRIP7MUXENABLE Register.....	833
Figure 9-49. TRIP8MUXENABLE Register.....	838
Figure 9-50. TRIP9MUXENABLE Register.....	843
Figure 9-51. TRIP10MUXENABLE Register.....	848
Figure 9-52. TRIP11MUXENABLE Register.....	853
Figure 9-53. TRIP12MUXENABLE Register.....	858
Figure 9-54. TRIPOUTINV Register.....	863
Figure 9-55. TRIPLOCK Register.....	865
Figure 9-56. OUTPUT1MUX0TO15CFG Register.....	868
Figure 9-57. OUTPUT1MUX16TO31CFG Register.....	871
Figure 9-58. OUTPUT2MUX0TO15CFG Register.....	874
Figure 9-59. OUTPUT2MUX16TO31CFG Register.....	877
Figure 9-60. OUTPUT3MUX0TO15CFG Register.....	880
Figure 9-61. OUTPUT3MUX16TO31CFG Register.....	883
Figure 9-62. OUTPUT4MUX0TO15CFG Register.....	886
Figure 9-63. OUTPUT4MUX16TO31CFG Register.....	889
Figure 9-64. OUTPUT5MUX0TO15CFG Register.....	892
Figure 9-65. OUTPUT5MUX16TO31CFG Register.....	895
Figure 9-66. OUTPUT6MUX0TO15CFG Register.....	898
Figure 9-67. OUTPUT6MUX16TO31CFG Register.....	901
Figure 9-68. OUTPUT7MUX0TO15CFG Register.....	904
Figure 9-69. OUTPUT7MUX16TO31CFG Register.....	907
Figure 9-70. OUTPUT8MUX0TO15CFG Register.....	910
Figure 9-71. OUTPUT8MUX16TO31CFG Register.....	913
Figure 9-72. OUTPUT1MUXENABLE Register.....	916
Figure 9-73. OUTPUT2MUXENABLE Register.....	921
Figure 9-74. OUTPUT3MUXENABLE Register.....	926
Figure 9-75. OUTPUT4MUXENABLE Register.....	931
Figure 9-76. OUTPUT5MUXENABLE Register.....	936
Figure 9-77. OUTPUT6MUXENABLE Register.....	941
Figure 9-78. OUTPUT7MUXENABLE Register.....	946
Figure 9-79. OUTPUT8MUXENABLE Register.....	951
Figure 9-80. OUTPUTLATCH Register.....	956
Figure 9-81. OUTPUTLATCHCLR Register.....	958
Figure 9-82. OUTPUTLATCHFRC Register.....	960
Figure 9-83. OUTPUTLATCHENABLE Register.....	962
Figure 9-84. OUTPUTINV Register.....	964
Figure 9-85. OUTPUTLOCK Register.....	966
Figure 10-1. Analog Subsystem Block Diagram.....	975
Figure 10-2. Analog Group Connections.....	976
Figure 10-3. EXTROSCCSR1 Register.....	983
Figure 10-4. INTERNALTESTCTL Register.....	984
Figure 10-5. CONFIGLOCK Register.....	986
Figure 10-6. TSNSCTL Register.....	987

Figure 10-7. ANAREFCTL Register.....	988
Figure 10-8. VMONCTL Register.....	989
Figure 10-9. CMPHPMXSEL Register.....	990
Figure 10-10. CMPLPMXSEL Register.....	991
Figure 10-11. CMPHNMXSEL Register.....	992
Figure 10-12. CMPLNMXSEL Register.....	993
Figure 10-13. ADCDACLOOPBACK Register.....	994
Figure 10-14. CMPSSCTL Register.....	995
Figure 10-15. LOCK Register.....	996
Figure 10-16. AGPIOCTRLA Register.....	998
Figure 10-17. AGPIOCTRLH Register.....	1000
Figure 11-1. ADC Module Block Diagram.....	1006
Figure 11-2. SOC Block Diagram.....	1010
Figure 11-3. Single-Ended Input Model.....	1012
Figure 11-4. Round Robin Priority Example.....	1017
Figure 11-5. High Priority Example.....	1018
Figure 11-6. Burst Priority Example.....	1020
Figure 11-7. ADC EOC Interrupts.....	1021
Figure 11-8. ADC PPB Block Diagram.....	1023
Figure 11-9. ADC PPB Interrupt Event.....	1025
Figure 11-10. Opens/Shorts Detection Circuit.....	1027
Figure 11-11. Input Circuit Equivalent with OSDETECT Enabled.....	1028
Figure 11-12. ADC Timings for 12-bit Mode in Early Interrupt Mode.....	1031
Figure 11-13. ADC Timings for 12-bit Mode in Late Interrupt Mode.....	1032
Figure 11-14. Example: Basic Synchronous Operation.....	1033
Figure 11-15. Example: Synchronous Operation with Multiple Trigger Sources.....	1034
Figure 11-16. Example: Synchronous Operation with Uneven SOC Numbers.....	1035
Figure 11-17. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow.....	1035
Figure 11-18. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions.....	1036
Figure 11-19. ADC Reference System.....	1038
Figure 11-20. CMPSS to ADC Loopback Connection.....	1039
Figure 11-21. ADCRESULT0 Register.....	1047
Figure 11-22. ADCRESULT1 Register.....	1048
Figure 11-23. ADCRESULT2 Register.....	1049
Figure 11-24. ADCRESULT3 Register.....	1050
Figure 11-25. ADCRESULT4 Register.....	1051
Figure 11-26. ADCRESULT5 Register.....	1052
Figure 11-27. ADCRESULT6 Register.....	1053
Figure 11-28. ADCRESULT7 Register.....	1054
Figure 11-29. ADCRESULT8 Register.....	1055
Figure 11-30. ADCRESULT9 Register.....	1056
Figure 11-31. ADCRESULT10 Register.....	1057
Figure 11-32. ADCRESULT11 Register.....	1058
Figure 11-33. ADCRESULT12 Register.....	1059
Figure 11-34. ADCRESULT13 Register.....	1060
Figure 11-35. ADCRESULT14 Register.....	1061
Figure 11-36. ADCRESULT15 Register.....	1062
Figure 11-37. ADCPPB1RESULT Register.....	1063
Figure 11-38. ADCPPB2RESULT Register.....	1064
Figure 11-39. ADCPPB3RESULT Register.....	1065
Figure 11-40. ADCPPB4RESULT Register.....	1066
Figure 11-41. ADCCTL1 Register.....	1070
Figure 11-42. ADCCTL2 Register.....	1072
Figure 11-43. ADCBURSTCTL Register.....	1073
Figure 11-44. ADCINTFLG Register.....	1075
Figure 11-45. ADCINTFLGCLR Register.....	1077
Figure 11-46. ADCINTOVF Register.....	1078
Figure 11-47. ADCINTOVFCLR Register.....	1079
Figure 11-48. ADCINTSEL1N2 Register.....	1080
Figure 11-49. ADCINTSEL3N4 Register.....	1082
Figure 11-50. ADCSOCPRICL Register.....	1084

Figure 11-51. ADCINTSOCSEL1 Register.....	1086
Figure 11-52. ADCINTSOCSEL2 Register.....	1088
Figure 11-53. ADCSOCFLG1 Register.....	1090
Figure 11-54. ADCSOCFRC1 Register.....	1094
Figure 11-55. ADCSOCOVF1 Register.....	1099
Figure 11-56. ADCSOCOVFCLR1 Register.....	1102
Figure 11-57. ADCSOC0CTL Register.....	1105
Figure 11-58. ADCSOC1CTL Register.....	1107
Figure 11-59. ADCSOC2CTL Register.....	1109
Figure 11-60. ADCSOC3CTL Register.....	1111
Figure 11-61. ADCSOC4CTL Register.....	1113
Figure 11-62. ADCSOC5CTL Register.....	1115
Figure 11-63. ADCSOC6CTL Register.....	1117
Figure 11-64. ADCSOC7CTL Register.....	1119
Figure 11-65. ADCSOC8CTL Register.....	1121
Figure 11-66. ADCSOC9CTL Register.....	1123
Figure 11-67. ADCSOC10CTL Register.....	1125
Figure 11-68. ADCSOC11CTL Register.....	1127
Figure 11-69. ADCSOC12CTL Register.....	1129
Figure 11-70. ADCSOC13CTL Register.....	1131
Figure 11-71. ADCSOC14CTL Register.....	1133
Figure 11-72. ADCSOC15CTL Register.....	1135
Figure 11-73. ADCEVTSTAT Register.....	1137
Figure 11-74. ADCEVTCLR Register.....	1140
Figure 11-75. ADCEVTSEL Register.....	1142
Figure 11-76. ADCEVTINTSEL Register.....	1144
Figure 11-77. ADCOSDETECT Register.....	1146
Figure 11-78. ADCCOUNTER Register.....	1147
Figure 11-79. ADCREV Register.....	1148
Figure 11-80. ADCOFFTRIM Register.....	1149
Figure 11-81. ADCPPB1CONFIG Register.....	1150
Figure 11-82. ADCPPB1STAMP Register.....	1152
Figure 11-83. ADCPPB1OFFCAL Register.....	1153
Figure 11-84. ADCPPB1OFFREF Register.....	1154
Figure 11-85. ADCPPB1TRIPHI Register.....	1155
Figure 11-86. ADCPPB1TRIPLO Register.....	1156
Figure 11-87. ADCPPB2CONFIG Register.....	1157
Figure 11-88. ADCPPB2STAMP Register.....	1159
Figure 11-89. ADCPPB2OFFCAL Register.....	1160
Figure 11-90. ADCPPB2OFFREF Register.....	1161
Figure 11-91. ADCPPB2TRIPHI Register.....	1162
Figure 11-92. ADCPPB2TRIPLO Register.....	1163
Figure 11-93. ADCPPB3CONFIG Register.....	1164
Figure 11-94. ADCPPB3STAMP Register.....	1166
Figure 11-95. ADCPPB3OFFCAL Register.....	1167
Figure 11-96. ADCPPB3OFFREF Register.....	1168
Figure 11-97. ADCPPB3TRIPHI Register.....	1169
Figure 11-98. ADCPPB3TRIPLO Register.....	1170
Figure 11-99. ADCPPB4CONFIG Register.....	1171
Figure 11-100. ADCPPB4STAMP Register.....	1173
Figure 11-101. ADCPPB4OFFCAL Register.....	1174
Figure 11-102. ADCPPB4OFFREF Register.....	1175
Figure 11-103. ADCPPB4TRIPHI Register.....	1176
Figure 11-104. ADCPPB4TRIPLO Register.....	1177
Figure 11-105. ADCINTCYCLE Register.....	1178
Figure 11-106. ADCINLTRIM2 Register.....	1179
Figure 11-107. ADCINLTRIM3 Register.....	1180
Figure 12-1. CMPSS Module Block Diagram.....	1187
Figure 12-2. CMPSS_LITE Module Block Diagram.....	1188
Figure 12-3. Comparator Block Diagram.....	1188
Figure 12-4. Reference DAC Block Diagram.....	1189

Figure 12-5. Ramp Generator Block Diagram.....	1191
Figure 12-6. Ramp Generator Behavior.....	1192
Figure 12-7. Digital Filter Behavior.....	1193
Figure 12-8. COMPCTL Register.....	1201
Figure 12-9. COMPHYSTL Register.....	1203
Figure 12-10. COMPSTS Register.....	1204
Figure 12-11. COMPSTCLR Register.....	1205
Figure 12-12. COMPDACHCTL Register.....	1206
Figure 12-13. DACHVALS Register.....	1208
Figure 12-14. DACHVALA Register.....	1209
Figure 12-15. RAMPHREFA Register.....	1210
Figure 12-16. RAMPHREFS Register.....	1211
Figure 12-17. RAMPHSTEPVALA Register.....	1212
Figure 12-18. RAMPHSTEPVALS Register.....	1213
Figure 12-19. RAMPHSTS Register.....	1214
Figure 12-20. DACLVALS Register.....	1215
Figure 12-21. DACLVALA Register.....	1216
Figure 12-22. RAMPHDLYA Register.....	1217
Figure 12-23. RAMPHDLYS Register.....	1218
Figure 12-24. CTRIPLFILCTL Register.....	1219
Figure 12-25. CTRIPLFILCLKCTL Register.....	1220
Figure 12-26. CTRIPHFILCTL Register.....	1221
Figure 12-27. CTRIPHFILCLKCTL Register.....	1222
Figure 12-28. COMPLOCK Register.....	1223
Figure 12-29. COMPCTL Register.....	1225
Figure 12-30. COMPHYSTL Register.....	1227
Figure 12-31. COMPSTS Register.....	1228
Figure 12-32. COMPSTCLR Register.....	1229
Figure 12-33. COMPDACHCTL Register.....	1230
Figure 12-34. DACHVALS Register.....	1231
Figure 12-35. DACHVALA Register.....	1232
Figure 12-36. DACLVALS Register.....	1233
Figure 12-37. DACLVALA Register.....	1234
Figure 12-38. CTRIPLFILCTL Register.....	1235
Figure 12-39. CTRIPLFILCLKCTL Register.....	1236
Figure 12-40. CTRIPHFILCTL Register.....	1237
Figure 12-41. CTRIPHFILCLKCTL Register.....	1238
Figure 12-42. COMPLOCK Register.....	1239
Figure 13-1. Multiple ePWM Modules.....	1247
Figure 13-2. Submodules and Signal Connections for an ePWM Module.....	1248
Figure 13-3. ePWM Modules and Critical Internal Signal Interconnects.....	1250
Figure 13-4. Time-Base Submodule.....	1253
Figure 13-5. Time-Base Submodule Signals and Registers.....	1254
Figure 13-6. Time-Base Frequency and Period.....	1256
Figure 13-7. Time-Base Counter Synchronization Scheme.....	1258
Figure 13-8. ePWM External SYNC Output.....	1259
Figure 13-9. Time-Base Up-Count Mode Waveforms.....	1262
Figure 13-10. Time-Base Down-Count Mode Waveforms.....	1263
Figure 13-11. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event.....	1264
Figure 13-12. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event.....	1265
Figure 13-13. Global Load: Signals and Registers.....	1266
Figure 13-14. One-Shot Sync Mode.....	1267
Figure 13-15. Counter-Compare Submodule.....	1268
Figure 13-16. Detailed View of the Counter-Compare Submodule.....	1269
Figure 13-17. Counter-Compare Event Waveforms in Up-Count Mode.....	1272
Figure 13-18. Counter-Compare Events in Down-Count Mode.....	1272
Figure 13-19. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event.....	1273
Figure 13-20. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event.....	1273
Figure 13-21. Action-Qualifier Submodule.....	1274

Figure 13-22. Action-Qualifier Submodule Inputs and Outputs.....	1275
Figure 13-23. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs.....	1276
Figure 13-24. AQCTL[SHDWAQAMODE].....	1279
Figure 13-25. AQCTL[SHDWAQBMODE].....	1279
Figure 13-26. Up-Down Count Mode Symmetrical Waveform.....	1281
Figure 13-27. Up, Single Edge Asymmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB— Active High.....	1282
Figure 13-28. Up, Single Edge Asymmetric Waveform with Independent Modulation on EPWMxA and EPWMxB— Active Low.....	1283
Figure 13-29. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA.....	1284
Figure 13-30. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Active Low.....	1284
Figure 13-31. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Complementary.....	1285
Figure 13-32. Up-Down Count, Dual-Edge Asymmetric Waveform, with Independent Modulation on EPWMxA—Active Low.....	1285
Figure 13-33. Up-Down Count, PWM Waveform Generation Utilizing T1 and T2 Events.....	1286
Figure 13-34. Dead_Band Submodule.....	1287
Figure 13-35. Configuration Options for the Dead-Band Submodule.....	1290
Figure 13-36. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%).....	1292
Figure 13-37. PWM Chopper Submodule.....	1294
Figure 13-38. PWM Chopper Submodule Operational Details.....	1295
Figure 13-39. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only.....	1295
Figure 13-40. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses.....	1296
Figure 13-41. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses.....	1297
Figure 13-42. Trip-Zone Submodule.....	1298
Figure 13-43. Trip-Zone Submodule Mode Control Logic.....	1302
Figure 13-44. Trip-Zone Submodule Interrupt Logic.....	1303
Figure 13-45. Event-Trigger Submodule.....	1304
Figure 13-46. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs.....	1305
Figure 13-47. Event-Trigger Interrupt Generator.....	1307
Figure 13-48. Event-Trigger SOCA Pulse Generator.....	1308
Figure 13-49. Event-Trigger SOCB Pulse Generator.....	1308
Figure 13-50. Digital-Compare Submodule High-Level Block Diagram.....	1309
Figure 13-51. GPIO MUX-to-Trip Input Connectivity.....	1310
Figure 13-52. DCxEVT1 Event Triggering.....	1313
Figure 13-53. DCxEVT2 Event Triggering.....	1314
Figure 13-54. Event Filtering.....	1315
Figure 13-55. Blanking Window Timing Diagram.....	1316
Figure 13-56. Valley Switching.....	1318
Figure 13-57. ePWM X-BAR.....	1319
Figure 13-58. Simplified ePWM Module.....	1320
Figure 13-59. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave.....	1321
Figure 13-60. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$ .....	1322
Figure 13-61. Buck Waveforms for Control of Four Buck Stages (Note: Only three bucks shown here).....	1323
Figure 13-62. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$ ).....	1324
Figure 13-63. Buck Waveforms for Control of Four Buck Stages (Note: $F_{PWM2} = F_{PWM1}$ ).....	1325
Figure 13-64. Control of Two Half-H Bridge Stages ( $F_{PWM2} = N \times F_{PWM1}$ ).....	1326
Figure 13-65. Half-H Bridge Waveforms for Control of Two Half-H Bridge Stages (Note: Here $F_{PWM2} = F_{PWM1}$ ).....	1327
Figure 13-66. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control.....	1328
Figure 13-67. 3-Phase Inverter Waveforms for Control of Dual 3-Phase Inverter Stages (Only One Inverter Shown).....	1329
Figure 13-68. Configuring Two PWM Modules for Phase Control.....	1330
Figure 13-69. Timing Waveforms Associated with Phase Control Between Two Modules.....	1331
Figure 13-70. Control of 3-Phase Interleaved DC/DC Converter.....	1332
Figure 13-71. 3-Phase Interleaved DC/DC Converter Waveforms for Control of 3-Phase Interleaved DC/DC Converter....	1333
Figure 13-72. Control of Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ ).....	1334
Figure 13-73. ZVS Full-H Bridge Waveforms.....	1335
Figure 13-74. Peak Current Mode Control of Buck Converter.....	1336
Figure 13-75. Peak Current Mode Control Waveforms for Control of Buck Converter.....	1336
Figure 13-76. Control of Two Resonant Converter Stages.....	1337
Figure 13-77. H-Bridge LLC Resonant Converter PWM Waveforms.....	1337

Figure 13-78. HRPWM Block Diagram.....	1339
Figure 13-79. Resolution Calculations for Conventionally Generated PWM.....	1340
Figure 13-80. Operating Logic Using MEP.....	1341
Figure 13-81. HRPWM Extension Registers and Memory Configuration.....	1342
Figure 13-82. HRPWM System Interface.....	1343
Figure 13-83. HRPWM and HRCAL Source Clock.....	1344
Figure 13-84. Required PWM Waveform for a Requested Duty = 40.5%.....	1347
Figure 13-85. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0).....	1350
Figure 13-86. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0).....	1351
Figure 13-87. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1).....	1351
Figure 13-88. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1).....	1351
Figure 13-89. Simple Buck Controlled Converter Using a Single PWM.....	1358
Figure 13-90. PWM Waveform Generated for Simple Buck Controlled Converter.....	1358
Figure 13-91. Simple Reconstruction Filter for a PWM-based DAC.....	1360
Figure 13-92. PWM Waveform Generated for the PWM DAC Function.....	1360
Figure 13-93. TBCTL Register.....	1376
Figure 13-94. TBCTL2 Register.....	1378
Figure 13-95. EPWMSYNCINSEL Register.....	1379
Figure 13-96. TBCTR Register.....	1380
Figure 13-97. TBSTS Register.....	1381
Figure 13-98. EPWMSYNCOUTEN Register.....	1382
Figure 13-99. TBCTL3 Register.....	1384
Figure 13-100. CMPCTL Register.....	1385
Figure 13-101. CMPCTL2 Register.....	1387
Figure 13-102. DBCTL Register.....	1389
Figure 13-103. DBCTL2 Register.....	1392
Figure 13-104. AQCTL Register.....	1393
Figure 13-105. AQTSRCSEL Register.....	1395
Figure 13-106. PCCTL Register.....	1396
Figure 13-107. VCAPCTL Register.....	1398
Figure 13-108. VCNTCFG Register.....	1400
Figure 13-109. HRCNFG Register.....	1402
Figure 13-110. HRPWR Register.....	1404
Figure 13-111. HRMSTEP Register.....	1405
Figure 13-112. HRCNFG2 Register.....	1406
Figure 13-113. HRPCTL Register.....	1407
Figure 13-114. TRREM Register.....	1409
Figure 13-115. GLDCTL Register.....	1410
Figure 13-116. GLDCFG Register.....	1412
Figure 13-117. EPWMXLINK Register.....	1414
Figure 13-118. AQCTLA Register.....	1416
Figure 13-119. AQCTLA2 Register.....	1418
Figure 13-120. AQCTLB Register.....	1419
Figure 13-121. AQCTLB2 Register.....	1421
Figure 13-122. AQSFRC Register.....	1422
Figure 13-123. AQCSFRC Register.....	1423
Figure 13-124. DBREDHR Register.....	1424
Figure 13-125. DBRED Register.....	1425
Figure 13-126. DBFEDHR Register.....	1426
Figure 13-127. DBFED Register.....	1427
Figure 13-128. TBPFS Register.....	1428
Figure 13-129. TBPRDHR Register.....	1429
Figure 13-130. TBPRD Register.....	1430
Figure 13-131. CMPA Register.....	1431
Figure 13-132. CMPB Register.....	1432
Figure 13-133. CMPC Register.....	1433
Figure 13-134. CMPD Register.....	1434
Figure 13-135. GLDCTL2 Register.....	1435
Figure 13-136. SWVDELVAL Register.....	1436
Figure 13-137. TZSEL Register.....	1437
Figure 13-138. TZDCSEL Register.....	1439

Figure 13-139. TZCTL Register.....	1440
Figure 13-140. TZCTL2 Register.....	1442
Figure 13-141. TZCTLDCA Register.....	1444
Figure 13-142. TZCTLDCB Register.....	1446
Figure 13-143. TZEINT Register.....	1448
Figure 13-144. TZFLG Register.....	1449
Figure 13-145. TZCBCFLG Register.....	1451
Figure 13-146. TZOSTFLG Register.....	1453
Figure 13-147. TZCLR Register.....	1455
Figure 13-148. TZCBCCLR Register.....	1457
Figure 13-149. TZOSTCLR Register.....	1458
Figure 13-150. TZFRC Register.....	1459
Figure 13-151. ETSEL Register.....	1460
Figure 13-152. ETPS Register.....	1463
Figure 13-153. ETFLG Register.....	1466
Figure 13-154. ETCLR Register.....	1467
Figure 13-155. ETFRC Register.....	1468
Figure 13-156. ETINTPS Register.....	1469
Figure 13-157. ETSOCPS Register.....	1470
Figure 13-158. ETCNTINITCTL Register.....	1472
Figure 13-159. ETCNTINIT Register.....	1473
Figure 13-160. DCTRIPSEL Register.....	1474
Figure 13-161. DCACTL Register.....	1476
Figure 13-162. DCBCTL Register.....	1478
Figure 13-163. DCFCTL Register.....	1480
Figure 13-164. DCCAPCTL Register.....	1482
Figure 13-165. DCFOFFSET Register.....	1484
Figure 13-166. DCFOFFSETCNT Register.....	1485
Figure 13-167. DCFWINDOW Register.....	1486
Figure 13-168. DCFWINDOWCNT Register.....	1487
Figure 13-169. BLANKPULSEMIXSEL Register.....	1488
Figure 13-170. DCCAP Register.....	1490
Figure 13-171. DCAHTRIPSEL Register.....	1491
Figure 13-172. DCALTRIPSEL Register.....	1493
Figure 13-173. DCBHTRIPSEL Register.....	1495
Figure 13-174. DCBLTRIPSEL Register.....	1497
Figure 13-175. EPWMLOCK Register.....	1499
Figure 13-176. HWVDELVAL Register.....	1501
Figure 13-177. VCNTVAL Register.....	1502
Figure 14-1. Capture and APWM Modes of Operation.....	1520
Figure 14-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode.....	1521
Figure 14-3. eCAP Block Diagram.....	1522
Figure 14-4. Event Prescale Control.....	1523
Figure 14-5. Prescale Function Waveforms.....	1523
Figure 14-6. Details of the Continuous/One-shot Block.....	1524
Figure 14-7. Details of the Counter and Synchronization Block.....	1525
Figure 14-8. eCAP Synchronization Scheme.....	1526
Figure 14-9. Interrupts in eCAP Module.....	1527
Figure 14-10. PWM Waveform Details Of APWM Mode Operation.....	1528
Figure 14-11. Time-Base Frequency and Period Calculation.....	1529
Figure 14-12. Capture Sequence for Absolute Time-stamp and Rising-Edge Detect.....	1530
Figure 14-13. Capture Sequence for Absolute Time-stamp with Rising- and Falling-Edge Detect.....	1531
Figure 14-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect.....	1532
Figure 14-15. Capture Sequence for Delta Mode Time-stamp with Rising- and Falling-Edge Detect.....	1533
Figure 14-16. PWM Waveform Details of APWM Mode Operation.....	1534
Figure 14-17. TSCTR Register.....	1537
Figure 14-18. CTRPHS Register.....	1538
Figure 14-19. CAP1 Register.....	1539
Figure 14-20. CAP2 Register.....	1540
Figure 14-21. CAP3 Register.....	1541
Figure 14-22. CAP4 Register.....	1542

Figure 14-23. ECCTL0 Register.....	1543
Figure 14-24. ECCTL1 Register.....	1544
Figure 14-25. ECCTL2 Register.....	1546
Figure 14-26. ECEINT Register.....	1548
Figure 14-27. ECFLG Register.....	1550
Figure 14-28. ECCLR Register.....	1552
Figure 14-29. ECFRC Register.....	1553
Figure 14-30. ECAPSYNCINSEL Register.....	1554
Figure 15-1. Optical Encoder Disk.....	1558
Figure 15-2. QEP Encoder Output Signal for Forward/Reverse Movement.....	1558
Figure 15-3. Index Pulse Example.....	1559
Figure 15-4. Using eQEP to Decode Signals from SinCos Transducer.....	1562
Figure 15-5. Functional Block Diagram of the eQEP Peripheral.....	1564
Figure 15-6. Functional Block Diagram of Decoder Unit.....	1566
Figure 15-7. Quadrature Decoder State Machine.....	1567
Figure 15-8. Quadrature-clock and Direction Decoding.....	1568
Figure 15-9. Position Counter Reset by Index Pulse for 1000-Line Encoder (QPOSMAX = 3999 or 0xF9F).....	1570
Figure 15-10. Position Counter Underflow/Overflow (QPOSMAX = 4).....	1571
Figure 15-11. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1).....	1573
Figure 15-12. Strobe Event Latch (QEPCTL[SEL] = 1).....	1573
Figure 15-13. Latching Position Counter on ADCSOCA/ADCSOCB Event.....	1574
Figure 15-14. eQEP Position-compare Unit.....	1575
Figure 15-15. eQEP Position-compare Event Generation Points.....	1576
Figure 15-16. eQEP Position-compare Sync Output Pulse Stretcher.....	1576
Figure 15-17. eQEP Edge Capture Unit.....	1578
Figure 15-18. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010).....	1579
Figure 15-19. eQEP Edge Capture Unit - Timing Details.....	1579
Figure 15-20. eQEP Watchdog Timer.....	1581
Figure 15-21. eQEP Unit Timer Base.....	1581
Figure 15-22. QMA Module Block Diagram.....	1582
Figure 15-23. QMA Mode-1.....	1583
Figure 15-24. QMA Mode-2.....	1584
Figure 15-25. eQEP Interrupt Generation.....	1585
Figure 15-26. QPOSCNT Register.....	1592
Figure 15-27. QPOSINIT Register.....	1593
Figure 15-28. QPOSMAX Register.....	1594
Figure 15-29. QPOSCMP Register.....	1595
Figure 15-30. QPOSILAT Register.....	1596
Figure 15-31. QPOSSLAT Register.....	1597
Figure 15-32. QPOSLAT Register.....	1598
Figure 15-33. QUTMR Register.....	1599
Figure 15-34. QUPRD Register.....	1600
Figure 15-35. QWDTMR Register.....	1601
Figure 15-36. QWDPRD Register.....	1602
Figure 15-37. QDECCTL Register.....	1603
Figure 15-38. QEPCTL Register.....	1605
Figure 15-39. QCAPCTL Register.....	1607
Figure 15-40. QPOSCTL Register.....	1608
Figure 15-41. QEINT Register.....	1609
Figure 15-42. QFLG Register.....	1611
Figure 15-43. QCLR Register.....	1613
Figure 15-44. QFRC Register.....	1615
Figure 15-45. QEPSTS Register.....	1617
Figure 15-46. QCTMR Register.....	1619
Figure 15-47. QCPRD Register.....	1620
Figure 15-48. QCTMRLAT Register.....	1621
Figure 15-49. QCPRDLAT Register.....	1622
Figure 15-50. REV Register.....	1623
Figure 15-51. QEPSTROBESEL Register.....	1624
Figure 15-52. QMACTRL Register.....	1625
Figure 15-53. QEPSRCSEL Register.....	1626

Figure 16-1. CAN Block Diagram.....	1633
Figure 16-2. Accessing Message Objects Through IFx Registers.....	1634
Figure 16-3. CAN_MUX.....	1639
Figure 16-4. CAN Core in Silent Mode.....	1640
Figure 16-5. CAN Core in Loopback Mode.....	1641
Figure 16-6. CAN Core in External Loopback Mode.....	1642
Figure 16-7. CAN Core in Loopback Combined with Silent Mode.....	1643
Figure 16-8. CAN Interrupt Topology 1.....	1645
Figure 16-9. CAN Interrupt Topology 2.....	1645
Figure 16-10. Initialization of a Transmit Object.....	1648
Figure 16-11. Initialization of a Single Receive Object for Data Frames.....	1648
Figure 16-12. Initialization of a Single Receive Object for Remote Frames.....	1649
Figure 16-13. CPU Handling of a FIFO Buffer (Interrupt Driven).....	1654
Figure 16-14. Bit Timing.....	1655
Figure 16-15. Propagation Time Segment.....	1656
Figure 16-16. Synchronization on Late and Early Edges.....	1658
Figure 16-17. Filtering of Short Dominant Spikes.....	1659
Figure 16-18. Structure of the CAN Core's CAN Protocol Controller.....	1661
Figure 16-19. Data Transfer Between IF1 / IF2 Registers and Message RAM.....	1665
Figure 16-20. Structure of a Message Object.....	1666
Figure 16-21. Message RAM Representation in Debug Mode.....	1670
Figure 16-22. CAN_CTL Register.....	1676
Figure 16-23. CAN_ES Register.....	1679
Figure 16-24. CAN_ERRC Register.....	1681
Figure 16-25. CAN_BTR Register.....	1682
Figure 16-26. CAN_INT Register.....	1684
Figure 16-27. CAN_TEST Register.....	1685
Figure 16-28. CAN_PERR Register.....	1687
Figure 16-29. CAN_RAM_INIT Register.....	1688
Figure 16-30. CAN_GLB_INT_EN Register.....	1689
Figure 16-31. CAN_GLB_INT_FLG Register.....	1690
Figure 16-32. CAN_GLB_INT_CLR Register.....	1691
Figure 16-33. CAN_ABOTR Register.....	1692
Figure 16-34. CAN_TXRQ_X Register.....	1693
Figure 16-35. CAN_TXRQ_21 Register.....	1694
Figure 16-36. CAN_NDAT_X Register.....	1695
Figure 16-37. CAN_NDAT_21 Register.....	1696
Figure 16-38. CAN_IPEN_X Register.....	1697
Figure 16-39. CAN_IPEN_21 Register.....	1698
Figure 16-40. CAN_MVAL_X Register.....	1699
Figure 16-41. CAN_MVAL_21 Register.....	1700
Figure 16-42. CAN_IP_MUX21 Register.....	1701
Figure 16-43. CAN_IF1CMD Register.....	1702
Figure 16-44. CAN_IF1MSK Register.....	1705
Figure 16-45. CAN_IF1ARB Register.....	1706
Figure 16-46. CAN_IF1MCTL Register.....	1708
Figure 16-47. CAN_IF1DATA Register.....	1710
Figure 16-48. CAN_IF1DATB Register.....	1711
Figure 16-49. CAN_IF2CMD Register.....	1712
Figure 16-50. CAN_IF2MSK Register.....	1715
Figure 16-51. CAN_IF2ARB Register.....	1716
Figure 16-52. CAN_IF2MCTL Register.....	1718
Figure 16-53. CAN_IF2DATA Register.....	1720
Figure 16-54. CAN_IF2DATB Register.....	1721
Figure 16-55. CAN_IF3MSK Register.....	1722
Figure 16-56. CAN_IF3ARB Register.....	1723
Figure 16-57. CAN_IF3MCTL Register.....	1724
Figure 16-58. CAN_IF3DATA Register.....	1726
Figure 16-59. CAN_IF3DATB Register.....	1727
Figure 16-60. CAN_IF3UPD Register.....	1728
Figure 17-1. Multiple I2C Modules Connected.....	1734

Figure 17-2. I2C Module Conceptual Block Diagram.....	1737
Figure 17-3. Clocking Diagram for the I2C Module.....	1737
Figure 17-4. Roles of the Clock Divide-Down Values (ICCL and ICCH).....	1738
Figure 17-5. Bit Transfer on the I2C bus.....	1739
Figure 17-6. I2C Slave TX / RX Flowchart.....	1741
Figure 17-7. I2C Master TX / RX Flowchart.....	1742
Figure 17-8. I2C Module START and STOP Conditions.....	1743
Figure 17-9. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown).....	1744
Figure 17-10. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMMDR).....	1745
Figure 17-11. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMMDR).....	1745
Figure 17-12. I2C Module Free Data Format (FDF = 1 in I2CMMDR).....	1746
Figure 17-13. Repeated START Condition (in This Case, 7-Bit Addressing Format).....	1746
Figure 17-14. Synchronization of Two I2C Clock Generators During Arbitration.....	1747
Figure 17-15. Arbitration Procedure Between Two Master-Transmitters.....	1748
Figure 17-16. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit.....	1749
Figure 17-17. Enable Paths of the I2C Interrupt Requests.....	1752
Figure 17-18. Backwards Compatibility Mode and Forward Compatibility Bit, Slave Transmitter.....	1753
Figure 17-19. I2C FIFO Interrupt.....	1754
Figure 17-20. I2COAR Register.....	1759
Figure 17-21. I2CIER Register.....	1760
Figure 17-22. I2CSTR Register.....	1761
Figure 17-23. I2CCLKL Register.....	1765
Figure 17-24. I2CCLKH Register.....	1766
Figure 17-25. I2CCNT Register.....	1767
Figure 17-26. I2CDRR Register.....	1768
Figure 17-27. I2CSAR Register.....	1769
Figure 17-28. I2CDXR Register.....	1770
Figure 17-29. I2CMMDR Register.....	1771
Figure 17-30. I2CISRC Register.....	1775
Figure 17-31. I2CEMDR Register.....	1776
Figure 17-32. I2CPSC Register.....	1777
Figure 17-33. I2CFFTX Register.....	1778
Figure 17-34. I2CFFRX Register.....	1780
Figure 18-1. SCI CPU Interface.....	1784
Figure 18-2. Serial Communications Interface (SCI) Module Block Diagram.....	1786
Figure 18-3. Typical SCI Data Frame Formats.....	1788
Figure 18-4. Idle-Line Multiprocessor Communication Format.....	1790
Figure 18-5. Double-Buffered WUT and TXSHF.....	1791
Figure 18-6. Address-Bit Multiprocessor Communication Format.....	1792
Figure 18-7. SCI Asynchronous Communications Format.....	1793
Figure 18-8. SCI RX Signals in Communication Modes.....	1794
Figure 18-9. SCI TX Signals in Communications Mode.....	1795
Figure 18-10. SCI FIFO Interrupt Flags and Enable Logic.....	1800
Figure 18-11. SCICCR Register.....	1806
Figure 18-12. SCICTL1 Register.....	1808
Figure 18-13. SCIHBAUD Register.....	1810
Figure 18-14. SCILBAUD Register.....	1811
Figure 18-15. SCICTL2 Register.....	1812
Figure 18-16. SCIRXST Register.....	1814
Figure 18-17. SCIRXEMU Register.....	1818
Figure 18-18. SCIRXBUF Register.....	1819
Figure 18-19. SCITXBUF Register.....	1821
Figure 18-20. SCIFFTX Register.....	1822
Figure 18-21. SCIFFRX Register.....	1824
Figure 18-22. SCIFFCT Register.....	1826
Figure 18-23. SCIPRI Register.....	1827
Figure 19-1. SPI CPU Interface.....	1833
Figure 19-2. SPI Interrupt Flags and Enable Logic Generation.....	1836
Figure 19-3. SPI Master/Slave Connection.....	1837
Figure 19-4. SPI Module Master Configuration.....	1839
Figure 19-5. SPI Module Slave Configuration.....	1840

Figure 19-6. SPICLK Signal Options.....	1843
Figure 19-7. SPI: SPICLK-LSPCLK Characteristic when (BRR + 1) is Odd, BRR > 3, and CLKPOLARITY = 1.....	1844
Figure 19-8. SPI 3-wire Master Mode.....	1845
Figure 19-9. SPI 3-wire Slave Mode.....	1846
Figure 19-10. Five Bits per Character.....	1849
Figure 19-11. SPI Digital Audio Receiver Configuration Using Two SPIs.....	1852
Figure 19-12. Standard Right-Justified Digital Audio Data Format.....	1852
Figure 19-13. SPICCR Register.....	1856
Figure 19-14. SPICTL Register.....	1858
Figure 19-15. SPISTS Register.....	1860
Figure 19-16. SPIBRR Register.....	1862
Figure 19-17. SPIRXEMU Register.....	1863
Figure 19-18. SPIRXBUF Register.....	1864
Figure 19-19. SPITXBUF Register.....	1865
Figure 19-20. SPIDAT Register.....	1866
Figure 19-21. SPIFFTX Register.....	1867
Figure 19-22. SPIFFRX Register.....	1869
Figure 19-23. SPIFFCT Register.....	1871
Figure 19-24. SPIPRI Register.....	1872
Figure 20-1. EPG Overview Block Diagram.....	1876
Figure 20-2. EPG Detailed Block Diagram.....	1877
Figure 20-3. EPG Clock Generator.....	1878
Figure 20-4. EPG Clock Stop.....	1879
Figure 20-5. EPG Signal Generator Detailed Overview.....	1881
Figure 20-6. EPG Peripheral Signal Muxing.....	1884
Figure 20-7. EPG Interrupt.....	1888
Figure 20-8. GCTL0 Register.....	1893
Figure 20-9. GCTL1 Register.....	1895
Figure 20-10. GCTL2 Register.....	1896
Figure 20-11. GCTL3 Register.....	1898
Figure 20-12. EPGLOCK Register.....	1902
Figure 20-13. EPGCOMMIT Register.....	1903
Figure 20-14. GINTSTS Register.....	1904
Figure 20-15. GINTEN Register.....	1905
Figure 20-16. GINTCLR Register.....	1906
Figure 20-17. GINTFRC Register.....	1907
Figure 20-18. CLKDIV0_CTL0 Register.....	1908
Figure 20-19. CLKDIV0_CLKOFFSET Register.....	1909
Figure 20-20. CLKDIV1_CTL0 Register.....	1910
Figure 20-21. CLKDIV1_CLKOFFSET Register.....	1911
Figure 20-22. SIGGEN0_CTL0 Register.....	1912
Figure 20-23. SIGGEN0_CTL1 Register.....	1914
Figure 20-24. SIGGEN0_DATA0 Register.....	1915
Figure 20-25. SIGGEN0_DATA1 Register.....	1916
Figure 20-26. SIGGEN0_DATA0_ACTIVE Register.....	1917
Figure 20-27. SIGGEN0_DATA1_ACTIVE Register.....	1918
Figure 20-28. EPGMXSEL0 Register.....	1920
Figure 20-29. EPGMXSELLOCK Register.....	1923
Figure 20-30. EPGMXSELCOMMIT Register.....	1924

## List of Tables

Table 1-1. C2000Ware Root Directories.....	54
Table 2-1. TMU Supported Instructions.....	59
Table 3-1. Access to EALLOW-Protected Registers.....	63
Table 3-2. Reset Signals.....	63
Table 3-3. PIE Channel Mapping.....	70
Table 3-4. CPU Interrupt Vectors.....	73
Table 3-5. PIE Interrupt Vectors.....	74
Table 3-6. ExtR Specifications.....	81
Table 3-7. Sample Total Error Calculation.....	81

Table 3-8. Total Error Example Values.....	82
Table 3-9. ALT Modes.....	85
Table 3-10. Clock Connections Sorted by Clock Domain.....	87
Table 3-11. Clock Connections Sorted by Module Name.....	88
Table 3-12. Clock Source (OSCCLK) Failure Detection.....	92
Table 3-13. Example Watchdog Key Sequences.....	96
Table 3-14. Effect of Clock-Gating Low-Power Modes on the Device.....	98
Table 3-15. Error Handling in Different Scenarios.....	103
Table 3-16. Mapping of ECC Bits in Read Data from ECC/Parity Address Map.....	104
Table 3-17. Mapping of Parity Bits in Read Data from ECC/Parity Address Map.....	104
Table 3-18. System Control Registers Impacted.....	106
Table 3-19. SYSCTRL Base Address Table.....	114
Table 3-20. ACCESS_PROTECTION_REGS Registers.....	115
Table 3-21. ACCESS_PROTECTION_REGS Access Type Codes.....	115
Table 3-22. MAVFLG Register Field Descriptions.....	116
Table 3-23. MAVSET Register Field Descriptions.....	117
Table 3-24. MAVCLR Register Field Descriptions.....	118
Table 3-25. MAVINTEN Register Field Descriptions.....	119
Table 3-26. MCPUFAVADDR Register Field Descriptions.....	120
Table 3-27. MCPUWRAVADDR Register Field Descriptions.....	121
Table 3-28. CLK_CFG_REGS Registers.....	122
Table 3-29. CLK_CFG_REGS Access Type Codes.....	122
Table 3-30. CLKCFGLOCK1 Register Field Descriptions.....	124
Table 3-31. CLKSRCCTL1 Register Field Descriptions.....	126
Table 3-32. CLKSRCCTL2 Register Field Descriptions.....	128
Table 3-33. CLKSRCCTL3 Register Field Descriptions.....	129
Table 3-34. SYSPLLCTL1 Register Field Descriptions.....	130
Table 3-35. SYSPLLMULT Register Field Descriptions.....	131
Table 3-36. SYSPLLSTS Register Field Descriptions.....	132
Table 3-37. SYSCLKDIVSEL Register Field Descriptions.....	133
Table 3-38. XCLKOUTDIVSEL Register Field Descriptions.....	134
Table 3-39. LOSPCP Register Field Descriptions.....	135
Table 3-40. MCDCCR Register Field Descriptions.....	136
Table 3-41. X1CNT Register Field Descriptions.....	138
Table 3-42. XTALCR Register Field Descriptions.....	139
Table 3-43. XTALCR2 Register Field Descriptions.....	140
Table 3-44. CLKFAILCFG Register Field Descriptions.....	141
Table 3-45. CPU_SYS_REGS Registers.....	142
Table 3-46. CPU_SYS_REGS Access Type Codes.....	142
Table 3-47. CPUSYSLOCK1 Register Field Descriptions.....	144
Table 3-48. CPUSYSLOCK2 Register Field Descriptions.....	147
Table 3-49. PIEVERRADDR Register Field Descriptions.....	149
Table 3-50. PCLKCR0 Register Field Descriptions.....	150
Table 3-51. PCLKCR2 Register Field Descriptions.....	152
Table 3-52. PCLKCR3 Register Field Descriptions.....	154
Table 3-53. PCLKCR4 Register Field Descriptions.....	155
Table 3-54. PCLKCR7 Register Field Descriptions.....	156
Table 3-55. PCLKCR8 Register Field Descriptions.....	157
Table 3-56. PCLKCR9 Register Field Descriptions.....	158
Table 3-57. PCLKCR10 Register Field Descriptions.....	159
Table 3-58. PCLKCR13 Register Field Descriptions.....	160
Table 3-59. PCLKCR14 Register Field Descriptions.....	161
Table 3-60. PCLKCR21 Register Field Descriptions.....	162
Table 3-61. PCLKCR27 Register Field Descriptions.....	163
Table 3-62. SIMRESET Register Field Descriptions.....	164
Table 3-63. LPMCR Register Field Descriptions.....	165
Table 3-64. GPIOLPMSEL0 Register Field Descriptions.....	166
Table 3-65. GPIOLPMSEL1 Register Field Descriptions.....	169
Table 3-66. TMR2CLKCTL Register Field Descriptions.....	172
Table 3-67. RESCCLR Register Field Descriptions.....	173
Table 3-68. RESC Register Field Descriptions.....	175

Table 3-69. USER_REG1_SYSRSn Register Field Descriptions.....	177
Table 3-70. USER_REG2_SYSRSn Register Field Descriptions.....	178
Table 3-71. USER_REG1_XRSn Register Field Descriptions.....	179
Table 3-72. USER_REG2_XRSn Register Field Descriptions.....	180
Table 3-73. USER_REG1_PORESETn Register Field Descriptions.....	181
Table 3-74. USER_REG2_PORESETn Register Field Descriptions.....	182
Table 3-75. USER_REG3_PORESETn Register Field Descriptions.....	183
Table 3-76. USER_REG4_PORESETn Register Field Descriptions.....	184
Table 3-77. CPUTIMER_REGS Registers.....	185
Table 3-78. CPUTIMER_REGS Access Type Codes.....	185
Table 3-79. TIM Register Field Descriptions.....	186
Table 3-80. PRD Register Field Descriptions.....	187
Table 3-81. TCR Register Field Descriptions.....	188
Table 3-82. TPR Register Field Descriptions.....	190
Table 3-83. TPRH Register Field Descriptions.....	191
Table 3-84. DEV_CFG_REGS Registers.....	192
Table 3-85. DEV_CFG_REGS Access Type Codes.....	192
Table 3-86. PARTIDL Register Field Descriptions.....	194
Table 3-87. PARTIDH Register Field Descriptions.....	195
Table 3-88. REVID Register Field Descriptions.....	196
Table 3-89. TRIMERRSTS Register Field Descriptions.....	197
Table 3-90. SOFTPRES2 Register Field Descriptions.....	198
Table 3-91. SOFTPRES3 Register Field Descriptions.....	200
Table 3-92. SOFTPRES4 Register Field Descriptions.....	201
Table 3-93. SOFTPRES7 Register Field Descriptions.....	202
Table 3-94. SOFTPRES8 Register Field Descriptions.....	203
Table 3-95. SOFTPRES9 Register Field Descriptions.....	204
Table 3-96. SOFTPRES10 Register Field Descriptions.....	205
Table 3-97. SOFTPRES13 Register Field Descriptions.....	206
Table 3-98. SOFTPRES14 Register Field Descriptions.....	207
Table 3-99. SOFTPRES21 Register Field Descriptions.....	208
Table 3-100. SOFTPRES27 Register Field Descriptions.....	209
Table 3-101. SOFTPRES28 Register Field Descriptions.....	210
Table 3-102. TAP_STATUS Register Field Descriptions.....	211
Table 3-103. ECAPTYPE Register Field Descriptions.....	212
Table 3-104. MEM_CFG_REGS Registers.....	213
Table 3-105. MEM_CFG_REGS Access Type Codes.....	213
Table 3-106. DxLOCK Register Field Descriptions.....	215
Table 3-107. DxCOMMIT Register Field Descriptions.....	216
Table 3-108. DxACCPROT0 Register Field Descriptions.....	217
Table 3-109. DxACCPROT1 Register Field Descriptions.....	218
Table 3-110. DxTEST Register Field Descriptions.....	219
Table 3-111. DxINIT Register Field Descriptions.....	220
Table 3-112. DxINITDONE Register Field Descriptions.....	221
Table 3-113. DxRAMTEST_LOCK Register Field Descriptions.....	222
Table 3-114. LSxLOCK Register Field Descriptions.....	223
Table 3-115. LSxCOMMIT Register Field Descriptions.....	224
Table 3-116. LSxACCPROT0 Register Field Descriptions.....	225
Table 3-117. LSxTEST Register Field Descriptions.....	226
Table 3-118. LSxINIT Register Field Descriptions.....	227
Table 3-119. LSxINITDONE Register Field Descriptions.....	228
Table 3-120. LSxRAMTEST_LOCK Register Field Descriptions.....	229
Table 3-121. ROM_LOCK Register Field Descriptions.....	230
Table 3-122. ROM_TEST Register Field Descriptions.....	231
Table 3-123. ROM_FORCE_ERROR Register Field Descriptions.....	232
Table 3-124. MEMORY_ERROR_REGS Registers.....	233
Table 3-125. MEMORY_ERROR_REGS Access Type Codes.....	233
Table 3-126. UCERRFLG Register Field Descriptions.....	234
Table 3-127. UCERRSET Register Field Descriptions.....	235
Table 3-128. UCERRCLR Register Field Descriptions.....	236
Table 3-129. UCCPUREADDR Register Field Descriptions.....	237

Table 3-130. FLUCERRSTATUS Register Field Descriptions.....	238
Table 3-131. FLCERRSTATUS Register Field Descriptions.....	239
Table 3-132. CERRFLG Register Field Descriptions.....	241
Table 3-133. CERRSET Register Field Descriptions.....	242
Table 3-134. CERRCLR Register Field Descriptions.....	243
Table 3-135. CCPUREADDR Register Field Descriptions.....	244
Table 3-136. CERRCNT Register Field Descriptions.....	245
Table 3-137. CERRTHRES Register Field Descriptions.....	246
Table 3-138. CEINTFLG Register Field Descriptions.....	247
Table 3-139. CEINTCLR Register Field Descriptions.....	248
Table 3-140. CEINTSET Register Field Descriptions.....	249
Table 3-141. CEINTEN Register Field Descriptions.....	250
Table 3-142. NMI_INTRUPT_REGS Registers.....	251
Table 3-143. NMI_INTRUPT_REGS Access Type Codes.....	251
Table 3-144. NMICFG Register Field Descriptions.....	252
Table 3-145. NMIFLG Register Field Descriptions.....	253
Table 3-146. NMIFLGCLR Register Field Descriptions.....	255
Table 3-147. NMIFLGFRC Register Field Descriptions.....	257
Table 3-148. NMIWDCNT Register Field Descriptions.....	258
Table 3-149. NMIWDPRD Register Field Descriptions.....	259
Table 3-150. NMISHDFLG Register Field Descriptions.....	260
Table 3-151. ERRORSTS Register Field Descriptions.....	262
Table 3-152. ERRORSTSCLR Register Field Descriptions.....	263
Table 3-153. ERRORSTSFRC Register Field Descriptions.....	264
Table 3-154. ERRORCTL Register Field Descriptions.....	265
Table 3-155. ERRORLOCK Register Field Descriptions.....	266
Table 3-156. PIE_CTRL_REGS Registers.....	267
Table 3-157. PIE_CTRL_REGS Access Type Codes.....	267
Table 3-158. PIECTRL Register Field Descriptions.....	269
Table 3-159. PIEACK Register Field Descriptions.....	270
Table 3-160. PIEIER1 Register Field Descriptions.....	271
Table 3-161. PIEIFR1 Register Field Descriptions.....	272
Table 3-162. PIEIER2 Register Field Descriptions.....	273
Table 3-163. PIEIFR2 Register Field Descriptions.....	274
Table 3-164. PIEIER3 Register Field Descriptions.....	275
Table 3-165. PIEIFR3 Register Field Descriptions.....	276
Table 3-166. PIEIER4 Register Field Descriptions.....	277
Table 3-167. PIEIFR4 Register Field Descriptions.....	278
Table 3-168. PIEIER5 Register Field Descriptions.....	279
Table 3-169. PIEIFR5 Register Field Descriptions.....	280
Table 3-170. PIEIER6 Register Field Descriptions.....	281
Table 3-171. PIEIFR6 Register Field Descriptions.....	282
Table 3-172. PIEIER7 Register Field Descriptions.....	283
Table 3-173. PIEIFR7 Register Field Descriptions.....	284
Table 3-174. PIEIER8 Register Field Descriptions.....	285
Table 3-175. PIEIFR8 Register Field Descriptions.....	286
Table 3-176. PIEIER9 Register Field Descriptions.....	287
Table 3-177. PIEIFR9 Register Field Descriptions.....	288
Table 3-178. PIEIER10 Register Field Descriptions.....	289
Table 3-179. PIEIFR10 Register Field Descriptions.....	290
Table 3-180. PIEIER11 Register Field Descriptions.....	291
Table 3-181. PIEIFR11 Register Field Descriptions.....	292
Table 3-182. PIEIER12 Register Field Descriptions.....	293
Table 3-183. PIEIFR12 Register Field Descriptions.....	294
Table 3-184. SYNC_SOC_REGS Registers.....	295
Table 3-185. SYNC_SOC_REGS Access Type Codes.....	295
Table 3-186. SYNCSELECT Register Field Descriptions.....	296
Table 3-187. ADCSOCOUTSELECT Register Field Descriptions.....	298
Table 3-188. SYNCLOCK Register Field Descriptions.....	300
Table 3-189. SYS_STATUS_REGS Registers.....	301
Table 3-190. SYS_STATUS_REGS Access Type Codes.....	301

Table 3-191. SYS_ERR_INT_FLG Register Field Descriptions.....	302
Table 3-192. SYS_ERR_INT_CLR Register Field Descriptions.....	304
Table 3-193. SYS_ERR_INT_SET Register Field Descriptions.....	306
Table 3-194. SYS_ERR_MASK Register Field Descriptions.....	308
Table 3-195. TEST_ERROR_REGS Registers.....	310
Table 3-196. TEST_ERROR_REGS Access Type Codes.....	310
Table 3-197. CPU_RAM_TEST_ERROR_STS Register Field Descriptions.....	311
Table 3-198. CPU_RAM_TEST_ERROR_STS_CLR Register Field Descriptions.....	312
Table 3-199. CPU_RAM_TEST_ERROR_ADDR Register Field Descriptions.....	313
Table 3-200. UID_REGS Registers.....	314
Table 3-201. UID_REGS Access Type Codes.....	314
Table 3-202. UID_PSRAND0 Register Field Descriptions.....	315
Table 3-203. UID_PSRAND1 Register Field Descriptions.....	316
Table 3-204. UID_PSRAND2 Register Field Descriptions.....	317
Table 3-205. UID_PSRAND3 Register Field Descriptions.....	318
Table 3-206. UID_PSRAND4 Register Field Descriptions.....	319
Table 3-207. UID_UNIQUE0 Register Field Descriptions.....	320
Table 3-208. UID_UNIQUE1 Register Field Descriptions.....	321
Table 3-209. UID_CHECKSUM Register Field Descriptions.....	322
Table 3-210. WD_REGS Registers.....	323
Table 3-211. WD_REGS Access Type Codes.....	323
Table 3-212. SCSR Register Field Descriptions.....	324
Table 3-213. WDCNTR Register Field Descriptions.....	325
Table 3-214. WDKEY Register Field Descriptions.....	326
Table 3-215. WDCR Register Field Descriptions.....	327
Table 3-216. WDWCR Register Field Descriptions.....	328
Table 3-217. XINT_REGS Registers.....	329
Table 3-218. XINT_REGS Access Type Codes.....	329
Table 3-219. XINT1CR Register Field Descriptions.....	330
Table 3-220. XINT2CR Register Field Descriptions.....	331
Table 3-221. XINT3CR Register Field Descriptions.....	332
Table 3-222. XINT4CR Register Field Descriptions.....	333
Table 3-223. XINT5CR Register Field Descriptions.....	334
Table 3-224. XINT1CTR Register Field Descriptions.....	335
Table 3-225. XINT2CTR Register Field Descriptions.....	336
Table 3-226. XINT3CTR Register Field Descriptions.....	337
Table 3-227. ASYSCTL Registers to Driverlib Functions.....	338
Table 3-228. CPUTIMER Registers to Driverlib Functions.....	339
Table 3-229. MEMCFG Registers to Driverlib Functions.....	339
Table 3-230. NMI Registers to Driverlib Functions.....	341
Table 3-231. PIE Registers to Driverlib Functions.....	342
Table 3-232. SYSCTL Registers to Driverlib Functions.....	344
Table 3-233. XINT Registers to Driverlib Functions.....	348
Table 4-1. Boot System Overview .....	352
Table 4-2. ROM Memory.....	352
Table 4-3. Device Boot ROM Sequence.....	353
Table 4-4. Device Default Boot Modes.....	353
Table 4-5. Custom Boot Modes.....	354
Table 4-6. BOOTPIN-CONFIG Bit Fields.....	355
Table 4-7. Standalone Boot Mode Select Pin Decoding.....	356
Table 4-8. BOOTDEF Bit Fields.....	357
Table 4-9. Zero Boot Pin Boot Table Result.....	358
Table 4-10. One Boot Pin Boot Table Result.....	358
Table 4-11. Three Boot Pins Boot Table Result.....	359
Table 4-12. Boot ROM Reset Causes and Actions.....	363
Table 4-13. Boot ROM Exceptions and Actions.....	363
Table 4-14. Boot ROM Registers.....	364
Table 4-15. DCSM Zone GPREG2 Bit Fields.....	365
Table 4-16. Flash Entry Point Addresses.....	365
Table 4-17. RAM Entry Point Address.....	366
Table 4-18. Secure Flash Entry Point Addresses.....	366

Table 4-19. Wait Boot Options.....	366
Table 4-20. Wait Point Addresses.....	366
Table 4-21. Secure Flash Boot Details.....	367
Table 4-22. Secure Flash Tag and Key Details.....	368
Table 4-23. Secure Flash Authentication Failure Actions.....	368
Table 4-24. Secure Flash on all CPUs Recommended Flow.....	368
Table 4-25. Boot ROM Memory Map.....	370
Table 4-26. Reserved RAM Memory Map.....	370
Table 4-27. ROM Symbol Tables.....	370
Table 4-28. Boot Mode Availability.....	371
Table 4-29. SPI 8-Bit Data Stream.....	373
Table 4-30. I2C 8-Bit Data Stream.....	378
Table 4-31. Parallel GPIO Boot 8-Bit Data Stream.....	379
Table 4-32. Bit-Rate Value for Internal Oscillators.....	383
Table 4-33. CAN 8-Bit Data Stream.....	384
Table 4-34. SCI Boot Options.....	385
Table 4-35. CAN Boot Options.....	385
Table 4-36. I2C Boot Options.....	385
Table 4-37. SPI Boot Options.....	386
Table 4-38. Parallel Boot Options.....	386
Table 4-39. Secure Copy Code Function.....	387
Table 4-40. Secure CRC Calculation Function.....	387
Table 4-41. CMAC Calculation Function.....	388
Table 4-42. CPU Boot Clock Sources.....	388
Table 4-43. CPU Clock State After Boot.....	388
Table 4-44. Boot Status Address.....	389
Table 4-45. Boot Status Bit Fields.....	389
Table 4-46. Boot Mode and MPOST Status Addresses.....	390
Table 4-47. Boot ROM Version Information.....	390
Table 4-48. LSB/MSB Loading Sequence in 8-Bit Data Stream.....	391
Table 4-49. Boot Loader Options.....	393
Table 5-1. RAM/Flash Status.....	397
Table 5-2. Security Levels.....	397
Table 5-3. Default Value of ZxOTP (Programmed by TI).....	398
Table 5-4. DCSM Base Address Table.....	412
Table 5-5. DCSM_Z1_REGS Registers.....	413
Table 5-6. DCSM_Z1_REGS Access Type Codes.....	413
Table 5-7. Z1_LINKPOINTER Register Field Descriptions.....	415
Table 5-8. Z1_OTPSECLOCK Register Field Descriptions.....	416
Table 5-9. Z1_JLM_ENABLE Register Field Descriptions.....	417
Table 5-10. Z1_LINKPOINTERERR Register Field Descriptions.....	418
Table 5-11. Z1_GPREG1 Register Field Descriptions.....	419
Table 5-12. Z1_GPREG2 Register Field Descriptions.....	420
Table 5-13. Z1_GPREG3 Register Field Descriptions.....	421
Table 5-14. Z1_GPREG4 Register Field Descriptions.....	422
Table 5-15. Z1_CSMKEY0 Register Field Descriptions.....	423
Table 5-16. Z1_CSMKEY1 Register Field Descriptions.....	424
Table 5-17. Z1_CSMKEY2 Register Field Descriptions.....	425
Table 5-18. Z1_CSMKEY3 Register Field Descriptions.....	426
Table 5-19. Z1_CR Register Field Descriptions.....	427
Table 5-20. Z1_GRABSECT1R Register Field Descriptions.....	429
Table 5-21. Z1_GRABSECT2R Register Field Descriptions.....	433
Table 5-22. Z1_GRABSECT3R Register Field Descriptions.....	437
Table 5-23. Z1_GRABRAM1R Register Field Descriptions.....	440
Table 5-24. Z1_EXEONLYSECT1R Register Field Descriptions.....	442
Table 5-25. Z1_EXEONLYSECT2R Register Field Descriptions.....	448
Table 5-26. Z1_EXEONLYRAM1R Register Field Descriptions.....	451
Table 5-27. Z1_JTAGKEY0 Register Field Descriptions.....	453
Table 5-28. Z1_JTAGKEY1 Register Field Descriptions.....	454
Table 5-29. Z1_JTAGKEY2 Register Field Descriptions.....	455
Table 5-30. Z1_JTAGKEY3 Register Field Descriptions.....	456

Table 5-31. Z1_CMACKKEY0 Register Field Descriptions.....	457
Table 5-32. Z1_CMACKKEY1 Register Field Descriptions.....	458
Table 5-33. Z1_CMACKKEY2 Register Field Descriptions.....	459
Table 5-34. Z1_CMACKKEY3 Register Field Descriptions.....	460
Table 5-35. DCSM_Z2_REGS Registers.....	461
Table 5-36. DCSM_Z2_REGS Access Type Codes.....	461
Table 5-37. Z2_LINKPOINTER Register Field Descriptions.....	462
Table 5-38. Z2_OTPSECLOCK Register Field Descriptions.....	463
Table 5-39. Z2_LINKPOINTERERR Register Field Descriptions.....	464
Table 5-40. Z2_GPREG1 Register Field Descriptions.....	465
Table 5-41. Z2_GPREG2 Register Field Descriptions.....	466
Table 5-42. Z2_GPREG3 Register Field Descriptions.....	467
Table 5-43. Z2_GPREG4 Register Field Descriptions.....	468
Table 5-44. Z2_CSMKEY0 Register Field Descriptions.....	469
Table 5-45. Z2_CSMKEY1 Register Field Descriptions.....	470
Table 5-46. Z2_CSMKEY2 Register Field Descriptions.....	471
Table 5-47. Z2_CSMKEY3 Register Field Descriptions.....	472
Table 5-48. Z2_CR Register Field Descriptions.....	473
Table 5-49. Z2_GRABSECT1R Register Field Descriptions.....	475
Table 5-50. Z2_GRABSECT2R Register Field Descriptions.....	479
Table 5-51. Z2_GRABSECT3R Register Field Descriptions.....	483
Table 5-52. Z2_GRABRAM1R Register Field Descriptions.....	486
Table 5-53. Z2_EXEONLYSECT1R Register Field Descriptions.....	488
Table 5-54. Z2_EXEONLYSECT2R Register Field Descriptions.....	494
Table 5-55. Z2_EXEONLYRAM1R Register Field Descriptions.....	497
Table 5-56. DCSM_COMMON_REGS Registers.....	499
Table 5-57. DCSM_COMMON_REGS Access Type Codes.....	499
Table 5-58. FLSEM Register Field Descriptions.....	500
Table 5-59. SECTSTAT1 Register Field Descriptions.....	501
Table 5-60. SECTSTAT2 Register Field Descriptions.....	504
Table 5-61. SECTSTAT3 Register Field Descriptions.....	507
Table 5-62. RAMSTAT1 Register Field Descriptions.....	510
Table 5-63. SECERRSTAT Register Field Descriptions.....	512
Table 5-64. SECERRCLR Register Field Descriptions.....	513
Table 5-65. SECERRFRC Register Field Descriptions.....	514
Table 5-66. DENYCODE Register Field Descriptions.....	515
Table 5-67. RAMOPENSTAT Register Field Descriptions.....	517
Table 5-68. RAMOPENFRC Register Field Descriptions.....	518
Table 5-69. RAMOPENCLR Register Field Descriptions.....	519
Table 5-70. RAMOPENLOCK Register Field Descriptions.....	520
Table 5-71. DCSM_Z1_OTP Registers.....	521
Table 5-72. DCSM_Z1_OTP Access Type Codes.....	521
Table 5-73. Z1OTP_LINKPOINTER1 Register Field Descriptions.....	522
Table 5-74. Z1OTP_LINKPOINTER2 Register Field Descriptions.....	523
Table 5-75. Z1OTP_LINKPOINTER3 Register Field Descriptions.....	524
Table 5-76. Z1OTP_JLM_ENABLE Register Field Descriptions.....	525
Table 5-77. Z1OTP_GPREG1 Register Field Descriptions.....	526
Table 5-78. Z1OTP_GPREG2 Register Field Descriptions.....	527
Table 5-79. Z1OTP_GPREG3 Register Field Descriptions.....	528
Table 5-80. Z1OTP_GPREG4 Register Field Descriptions.....	529
Table 5-81. Z1OTP_PSWDLOCK Register Field Descriptions.....	530
Table 5-82. Z1OTP_JTAGPSWDH0 Register Field Descriptions.....	531
Table 5-83. Z1OTP_JTAGPSWDH1 Register Field Descriptions.....	532
Table 5-84. Z1OTP_CMACKKEY0 Register Field Descriptions.....	533
Table 5-85. Z1OTP_CMACKKEY1 Register Field Descriptions.....	534
Table 5-86. Z1OTP_CMACKKEY2 Register Field Descriptions.....	535
Table 5-87. Z1OTP_CMACKKEY3 Register Field Descriptions.....	536
Table 5-88. DCSM_Z2_OTP Registers.....	537
Table 5-89. DCSM_Z2_OTP Access Type Codes.....	537
Table 5-90. Z2OTP_LINKPOINTER1 Register Field Descriptions.....	538
Table 5-91. Z2OTP_LINKPOINTER2 Register Field Descriptions.....	539

Table 5-92. Z2OTP_LINKPOINTER3 Register Field Descriptions.....	540
Table 5-93. Z2OTP_GPREG1 Register Field Descriptions.....	541
Table 5-94. Z2OTP_GPREG2 Register Field Descriptions.....	542
Table 5-95. Z2OTP_GPREG3 Register Field Descriptions.....	543
Table 5-96. Z2OTP_GPREG4 Register Field Descriptions.....	544
Table 5-97. Z2OTP_PSWDLOCK Register Field Descriptions.....	545
Table 5-98. DCSM Registers to Driverlib Functions.....	545
Table 6-1. FLASH Base Address Table.....	566
Table 6-2. FLASH_CTRL_REGS Registers.....	567
Table 6-3. FLASH_CTRL_REGS Access Type Codes.....	567
Table 6-4. FRDCNTL Register Field Descriptions.....	568
Table 6-5. FLPROT Register Field Descriptions.....	569
Table 6-6. FRD_INTF_CTRL Register Field Descriptions.....	570
Table 6-7. FLASH_ECC_REGS Registers.....	571
Table 6-8. FLASH_ECC_REGS Access Type Codes.....	571
Table 6-9. ECC_ENABLE Register Field Descriptions.....	572
Table 6-10. FECC_CTRL Register Field Descriptions.....	573
Table 6-11. FLASH Registers to Driverlib Functions.....	573
Table 7-1. DCC Base Address Table.....	586
Table 7-2. DCC_REGS Registers.....	587
Table 7-3. DCC_REGS Access Type Codes.....	587
Table 7-4. DCCGCTRL Register Field Descriptions.....	588
Table 7-5. DCCCNTSEED0 Register Field Descriptions.....	589
Table 7-6. DCCVALIDSEED0 Register Field Descriptions.....	590
Table 7-7. DCCCNTSEED1 Register Field Descriptions.....	591
Table 7-8. DCCSTATUS Register Field Descriptions.....	592
Table 7-9. DCCCNT0 Register Field Descriptions.....	593
Table 7-10. DCCVALID0 Register Field Descriptions.....	594
Table 7-11. DCCCNT1 Register Field Descriptions.....	595
Table 7-12. DCCCLKSRC1 Register Field Descriptions.....	596
Table 7-13. DCCCLKSRC0 Register Field Descriptions.....	598
Table 7-14. DCC Registers to Driverlib Functions.....	598
Table 8-1. AGPIO Configuration.....	605
Table 8-2. Sampling Period.....	608
Table 8-3. Sampling Frequency.....	608
Table 8-4. Case 1: Three-Sample Sampling-Window Width.....	609
Table 8-5. Case 2: Six-Sample Sampling-Window Width.....	609
Table 8-6. GPIO Muxed Pins.....	611
Table 8-7. GPIO and Peripheral Muxing.....	613
Table 8-8. Peripheral Muxing (Multiple Pins Assigned).....	614
Table 8-9. GPIO Base Address Table.....	616
Table 8-10. GPIO_CTRL_REGS Registers.....	617
Table 8-11. GPIO_CTRL_REGS Access Type Codes.....	618
Table 8-12. GPACTRL Register Field Descriptions.....	619
Table 8-13. GPAQSEL1 Register Field Descriptions.....	620
Table 8-14. GPAQSEL2 Register Field Descriptions.....	622
Table 8-15. GPAMUX1 Register Field Descriptions.....	624
Table 8-16. GPAMUX2 Register Field Descriptions.....	625
Table 8-17. GPADIR Register Field Descriptions.....	626
Table 8-18. GPAPUD Register Field Descriptions.....	628
Table 8-19. GPAINV Register Field Descriptions.....	630
Table 8-20. GPAODR Register Field Descriptions.....	632
Table 8-21. GPAAMSEL Register Field Descriptions.....	634
Table 8-22. GPAGMUX1 Register Field Descriptions.....	636
Table 8-23. GPAGMUX2 Register Field Descriptions.....	637
Table 8-24. GPALOCK Register Field Descriptions.....	638
Table 8-25. GPACR Register Field Descriptions.....	640
Table 8-26. GPBCTRL Register Field Descriptions.....	642
Table 8-27. GPBQSEL1 Register Field Descriptions.....	643
Table 8-28. GPBMUX1 Register Field Descriptions.....	645
Table 8-29. GPBDIR Register Field Descriptions.....	646

Table 8-30. GPBPUD Register Field Descriptions.....	648
Table 8-31. GPBINV Register Field Descriptions.....	650
Table 8-32. GPBODR Register Field Descriptions.....	652
Table 8-33. GPBGMUX1 Register Field Descriptions.....	654
Table 8-34. GPBLOCK Register Field Descriptions.....	655
Table 8-35. GPBCR Register Field Descriptions.....	657
Table 8-36. GPHCTRL Register Field Descriptions.....	659
Table 8-37. GPHQSEL1 Register Field Descriptions.....	660
Table 8-38. GPHQSEL2 Register Field Descriptions.....	662
Table 8-39. GPHMUX1 Register Field Descriptions.....	664
Table 8-40. GPHMUX2 Register Field Descriptions.....	665
Table 8-41. GPHDIR Register Field Descriptions.....	666
Table 8-42. GPHPUD Register Field Descriptions.....	668
Table 8-43. GPHINV Register Field Descriptions.....	672
Table 8-44. GPHODR Register Field Descriptions.....	675
Table 8-45. GPHAMSEL Register Field Descriptions.....	677
Table 8-46. GPHGMUX1 Register Field Descriptions.....	681
Table 8-47. GPHGMUX2 Register Field Descriptions.....	682
Table 8-48. GPHLOCK Register Field Descriptions.....	683
Table 8-49. GPHCR Register Field Descriptions.....	686
Table 8-50. GPIO_DATA_REGS Registers.....	689
Table 8-51. GPIO_DATA_REGS Access Type Codes.....	689
Table 8-52. GPADAT Register Field Descriptions.....	690
Table 8-53. GPASET Register Field Descriptions.....	692
Table 8-54. GPACLEAR Register Field Descriptions.....	694
Table 8-55. GPATOGGLE Register Field Descriptions.....	696
Table 8-56. GPBDAT Register Field Descriptions.....	698
Table 8-57. GPBSET Register Field Descriptions.....	700
Table 8-58. GPBCLEAR Register Field Descriptions.....	702
Table 8-59. GPBTOGGLE Register Field Descriptions.....	704
Table 8-60. GPHDAT Register Field Descriptions.....	706
Table 8-61. GPHSET Register Field Descriptions.....	711
Table 8-62. GPHCLEAR Register Field Descriptions.....	713
Table 8-63. GPHTOGGLE Register Field Descriptions.....	715
Table 8-64. GPIO_DATA_READ_REGS Registers.....	717
Table 8-65. GPIO_DATA_READ_REGS Access Type Codes.....	717
Table 8-66. GPADAT_R Register Field Descriptions.....	718
Table 8-67. GPBDAT_R Register Field Descriptions.....	719
Table 8-68. GPHDAT_R Register Field Descriptions.....	720
Table 8-69. GPIO Registers to Driverlib Functions.....	720
Table 9-1. Input X-BAR Destinations.....	726
Table 9-2. EPWM X-BAR Mux Configuration Table.....	728
Table 9-3. Output X-BAR Mux Configuration Table.....	730
Table 9-4. XBAR Base Address Table.....	732
Table 9-5. INPUT_XBAR_REGS Registers.....	733
Table 9-6. INPUT_XBAR_REGS Access Type Codes.....	733
Table 9-7. INPUT1SELECT Register Field Descriptions.....	734
Table 9-8. INPUT2SELECT Register Field Descriptions.....	735
Table 9-9. INPUT3SELECT Register Field Descriptions.....	736
Table 9-10. INPUT4SELECT Register Field Descriptions.....	737
Table 9-11. INPUT5SELECT Register Field Descriptions.....	738
Table 9-12. INPUT6SELECT Register Field Descriptions.....	739
Table 9-13. INPUT7SELECT Register Field Descriptions.....	740
Table 9-14. INPUT8SELECT Register Field Descriptions.....	741
Table 9-15. INPUT9SELECT Register Field Descriptions.....	742
Table 9-16. INPUT10SELECT Register Field Descriptions.....	743
Table 9-17. INPUT11SELECT Register Field Descriptions.....	744
Table 9-18. INPUT12SELECT Register Field Descriptions.....	745
Table 9-19. INPUT13SELECT Register Field Descriptions.....	746
Table 9-20. INPUT14SELECT Register Field Descriptions.....	747
Table 9-21. INPUT15SELECT Register Field Descriptions.....	748

Table 9-22. INPUT16SELECT Register Field Descriptions.....	749
Table 9-23. INPUTSELECTLOCK Register Field Descriptions.....	750
Table 9-24. XBAR_REGS Registers.....	752
Table 9-25. XBAR_REGS Access Type Codes.....	752
Table 9-26. XBARFLG1 Register Field Descriptions.....	753
Table 9-27. XBARFLG2 Register Field Descriptions.....	756
Table 9-28. XBARFLG3 Register Field Descriptions.....	760
Table 9-29. XBARFLG4 Register Field Descriptions.....	762
Table 9-30. XBARCLR1 Register Field Descriptions.....	764
Table 9-31. XBARCLR2 Register Field Descriptions.....	766
Table 9-32. XBARCLR3 Register Field Descriptions.....	769
Table 9-33. XBARCLR4 Register Field Descriptions.....	771
Table 9-34. EPWM_XBAR_REGS Registers.....	773
Table 9-35. EPWM_XBAR_REGS Access Type Codes.....	773
Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions.....	775
Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions.....	778
Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions.....	781
Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions.....	784
Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions.....	787
Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions.....	790
Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions.....	793
Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions.....	796
Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions.....	799
Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions.....	802
Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions.....	805
Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions.....	808
Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions.....	811
Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions.....	814
Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions.....	817
Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions.....	820
Table 9-52. TRIP4MUXENABLE Register Field Descriptions.....	823
Table 9-53. TRIP5MUXENABLE Register Field Descriptions.....	828
Table 9-54. TRIP7MUXENABLE Register Field Descriptions.....	833
Table 9-55. TRIP8MUXENABLE Register Field Descriptions.....	838
Table 9-56. TRIP9MUXENABLE Register Field Descriptions.....	843
Table 9-57. TRIP10MUXENABLE Register Field Descriptions.....	848
Table 9-58. TRIP11MUXENABLE Register Field Descriptions.....	853
Table 9-59. TRIP12MUXENABLE Register Field Descriptions.....	858
Table 9-60. TRIPOUTINV Register Field Descriptions.....	863
Table 9-61. TRIPLOCK Register Field Descriptions.....	865
Table 9-62. OUTPUT_XBAR_REGS Registers.....	866
Table 9-63. OUTPUT_XBAR_REGS Access Type Codes.....	866
Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions.....	868
Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions.....	871
Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions.....	874
Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions.....	877
Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions.....	880
Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions.....	883
Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions.....	886
Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions.....	889
Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions.....	892
Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions.....	895
Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions.....	898
Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions.....	901
Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions.....	904
Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions.....	907
Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions.....	910
Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions.....	913
Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions.....	916
Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions.....	921
Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions.....	926

Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions.....	931
Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions.....	936
Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions.....	941
Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions.....	946
Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions.....	951
Table 9-88. OUTPUTLATCH Register Field Descriptions.....	956
Table 9-89. OUTPUTLATCHCLR Register Field Descriptions.....	958
Table 9-90. OUTPUTLATCHFRC Register Field Descriptions.....	960
Table 9-91. OUTPUTLATCHENABLE Register Field Descriptions.....	962
Table 9-92. OUTPUTINV Register Field Descriptions.....	964
Table 9-93. OUTPUTLOCK Register Field Descriptions.....	966
Table 9-94. INPUTXBAR Registers to Driverlib Functions.....	967
Table 9-95. XBAR Registers to Driverlib Functions.....	967
Table 9-96. EPWMXBAR Registers to Driverlib Functions.....	968
Table 9-97. OUTPUTXBAR Registers to Driverlib Functions.....	969
Table 10-1. AGPIO Configuration.....	977
Table 10-2. Analog Pins and Internal Connections.....	978
Table 10-3. Analog Signal Descriptions.....	979
Table 10-4. Reference Summary.....	979
Table 10-5. ASBSYS Base Address Table.....	980
Table 10-6. ANALOG_SUBSYS_REGS Registers.....	981
Table 10-7. ANALOG_SUBSYS_REGS Access Type Codes.....	981
Table 10-8. EXTROSCCSR1 Register Field Descriptions.....	983
Table 10-9. INTERNALTESTCTL Register Field Descriptions.....	984
Table 10-10. CONFIGLOCK Register Field Descriptions.....	986
Table 10-11. TSNCTL Register Field Descriptions.....	987
Table 10-12. ANAREFCTL Register Field Descriptions.....	988
Table 10-13. VMONCTL Register Field Descriptions.....	989
Table 10-14. CMPHPMXSEL Register Field Descriptions.....	990
Table 10-15. CMPLPMXSEL Register Field Descriptions.....	991
Table 10-16. CMPHNMXSEL Register Field Descriptions.....	992
Table 10-17. CmplNMXSEL Register Field Descriptions.....	993
Table 10-18. ADCDACLOOPBACK Register Field Descriptions.....	994
Table 10-19. CMPSSCTL Register Field Descriptions.....	995
Table 10-20. LOCK Register Field Descriptions.....	996
Table 10-21. AGPIOCTRLA Register Field Descriptions.....	998
Table 10-22. AGPIOCTRLH Register Field Descriptions.....	1000
Table 11-1. ADC Options and Configuration Levels.....	1007
Table 11-2. Analog to 12-bit Digital Formulas.....	1009
Table 11-3. 12-Bit Digital-to-Analog Formulas.....	1009
Table 11-4. ADC SOC Trigger Selection.....	1011
Table 11-5. Channel Selection of Input Pins.....	1013
Table 11-6. Example Requirements for Multiple Signal Sampling.....	1015
Table 11-7. Example Connections for Multiple Signal Sampling.....	1015
Table 11-8. DETECTCFG Settings.....	1027
Table 11-9. ADC Timing Parameter Descriptions.....	1030
Table 11-10. ADC Timings in 12-bit Mode.....	1032
Table 11-11. ADC Base Address Table.....	1045
Table 11-12. ADC_RESULT_REGS Registers.....	1046
Table 11-13. ADC_RESULT_REGS Access Type Codes.....	1046
Table 11-14. ADCRESULT0 Register Field Descriptions.....	1047
Table 11-15. ADCRESULT1 Register Field Descriptions.....	1048
Table 11-16. ADCRESULT2 Register Field Descriptions.....	1049
Table 11-17. ADCRESULT3 Register Field Descriptions.....	1050
Table 11-18. ADCRESULT4 Register Field Descriptions.....	1051
Table 11-19. ADCRESULT5 Register Field Descriptions.....	1052
Table 11-20. ADCRESULT6 Register Field Descriptions.....	1053
Table 11-21. ADCRESULT7 Register Field Descriptions.....	1054
Table 11-22. ADCRESULT8 Register Field Descriptions.....	1055
Table 11-23. ADCRESULT9 Register Field Descriptions.....	1056
Table 11-24. ADCRESULT10 Register Field Descriptions.....	1057

Table 11-25. ADCRESULT11 Register Field Descriptions.....	1058
Table 11-26. ADCRESULT12 Register Field Descriptions.....	1059
Table 11-27. ADCRESULT13 Register Field Descriptions.....	1060
Table 11-28. ADCRESULT14 Register Field Descriptions.....	1061
Table 11-29. ADCRESULT15 Register Field Descriptions.....	1062
Table 11-30. ADCPPB1RESULT Register Field Descriptions.....	1063
Table 11-31. ADCPPB2RESULT Register Field Descriptions.....	1064
Table 11-32. ADCPPB3RESULT Register Field Descriptions.....	1065
Table 11-33. ADCPPB4RESULT Register Field Descriptions.....	1066
Table 11-34. ADC_REGS Registers.....	1067
Table 11-35. ADC_REGS Access Type Codes.....	1068
Table 11-36. ADCCTL1 Register Field Descriptions.....	1070
Table 11-37. ADCCTL2 Register Field Descriptions.....	1072
Table 11-38. ADCBURSTCTL Register Field Descriptions.....	1073
Table 11-39. ADCINTFLG Register Field Descriptions.....	1075
Table 11-40. ADCINTFLGCLR Register Field Descriptions.....	1077
Table 11-41. ADCINTOVF Register Field Descriptions.....	1078
Table 11-42. ADCINTOVFCLR Register Field Descriptions.....	1079
Table 11-43. ADCINTSEL1N2 Register Field Descriptions.....	1080
Table 11-44. ADCINTSEL3N4 Register Field Descriptions.....	1082
Table 11-45. ADCSOCPRCTL Register Field Descriptions.....	1084
Table 11-46. ADCINTSOCSEL1 Register Field Descriptions.....	1086
Table 11-47. ADCINTSOCSEL2 Register Field Descriptions.....	1088
Table 11-48. ADCSOCFLG1 Register Field Descriptions.....	1090
Table 11-49. ADCSOCFRC1 Register Field Descriptions.....	1094
Table 11-50. ADCSOCOVF1 Register Field Descriptions.....	1099
Table 11-51. ADCSOCOVFCLR1 Register Field Descriptions.....	1102
Table 11-52. ADCSOC0CTL Register Field Descriptions.....	1105
Table 11-53. ADCSOC1CTL Register Field Descriptions.....	1107
Table 11-54. ADCSOC2CTL Register Field Descriptions.....	1109
Table 11-55. ADCSOC3CTL Register Field Descriptions.....	1111
Table 11-56. ADCSOC4CTL Register Field Descriptions.....	1113
Table 11-57. ADCSOC5CTL Register Field Descriptions.....	1115
Table 11-58. ADCSOC6CTL Register Field Descriptions.....	1117
Table 11-59. ADCSOC7CTL Register Field Descriptions.....	1119
Table 11-60. ADCSOC8CTL Register Field Descriptions.....	1121
Table 11-61. ADCSOC9CTL Register Field Descriptions.....	1123
Table 11-62. ADCSOC10CTL Register Field Descriptions.....	1125
Table 11-63. ADCSOC11CTL Register Field Descriptions.....	1127
Table 11-64. ADCSOC12CTL Register Field Descriptions.....	1129
Table 11-65. ADCSOC13CTL Register Field Descriptions.....	1131
Table 11-66. ADCSOC14CTL Register Field Descriptions.....	1133
Table 11-67. ADCSOC15CTL Register Field Descriptions.....	1135
Table 11-68. ADCEVTSTAT Register Field Descriptions.....	1137
Table 11-69. ADCEVTCLR Register Field Descriptions.....	1140
Table 11-70. ADCEVTSEL Register Field Descriptions.....	1142
Table 11-71. ADCEVTINTSEL Register Field Descriptions.....	1144
Table 11-72. ADCOSDETECT Register Field Descriptions.....	1146
Table 11-73. ADCCOUNTER Register Field Descriptions.....	1147
Table 11-74. ADCREV Register Field Descriptions.....	1148
Table 11-75. ADCOFFTRIM Register Field Descriptions.....	1149
Table 11-76. ADCPPB1CONFIG Register Field Descriptions.....	1150
Table 11-77. ADCPPB1STAMP Register Field Descriptions.....	1152
Table 11-78. ADCPPB1OFFCAL Register Field Descriptions.....	1153
Table 11-79. ADCPPB1OFFREF Register Field Descriptions.....	1154
Table 11-80. ADCPPB1TRIPHI Register Field Descriptions.....	1155
Table 11-81. ADCPPB1TRIPLO Register Field Descriptions.....	1156
Table 11-82. ADCPPB2CONFIG Register Field Descriptions.....	1157
Table 11-83. ADCPPB2STAMP Register Field Descriptions.....	1159
Table 11-84. ADCPPB2OFFCAL Register Field Descriptions.....	1160
Table 11-85. ADCPPB2OFFREF Register Field Descriptions.....	1161

Table 11-86. ADCPPB2TRIPHI Register Field Descriptions.....	1162
Table 11-87. ADCPPB2TRIPLO Register Field Descriptions.....	1163
Table 11-88. ADCPPB3CONFIG Register Field Descriptions.....	1164
Table 11-89. ADCPPB3STAMP Register Field Descriptions.....	1166
Table 11-90. ADCPPB3OFFCAL Register Field Descriptions.....	1167
Table 11-91. ADCPPB3OFFREF Register Field Descriptions.....	1168
Table 11-92. ADCPPB3TRIPHI Register Field Descriptions.....	1169
Table 11-93. ADCPPB3TRIPLO Register Field Descriptions.....	1170
Table 11-94. ADCPPB4CONFIG Register Field Descriptions.....	1171
Table 11-95. ADCPPB4STAMP Register Field Descriptions.....	1173
Table 11-96. ADCPPB4OFFCAL Register Field Descriptions.....	1174
Table 11-97. ADCPPB4OFFREF Register Field Descriptions.....	1175
Table 11-98. ADCPPB4TRIPHI Register Field Descriptions.....	1176
Table 11-99. ADCPPB4TRIPLO Register Field Descriptions.....	1177
Table 11-100. ADCINTCYCLE Register Field Descriptions.....	1178
Table 11-101. ADCINLTRIM2 Register Field Descriptions.....	1179
Table 11-102. ADCINLTRIM3 Register Field Descriptions.....	1180
Table 11-103. ADC Registers to Driverlib Functions.....	1180
Table 12-1. CMPSS and CMPSS_LITE Feature Comparison.....	1187
Table 12-2. CMPSS Base Address Table.....	1198
Table 12-3. CMPSS_REGS Registers.....	1199
Table 12-4. CMPSS_REGS Access Type Codes.....	1199
Table 12-5. COMPCTL Register Field Descriptions.....	1201
Table 12-6. COMPHYCTL Register Field Descriptions.....	1203
Table 12-7. COMPSTS Register Field Descriptions.....	1204
Table 12-8. COMPSTCLR Register Field Descriptions.....	1205
Table 12-9. COMPDACHCTL Register Field Descriptions.....	1206
Table 12-10. DACHVALS Register Field Descriptions.....	1208
Table 12-11. DACHVALA Register Field Descriptions.....	1209
Table 12-12. RAMPHREFA Register Field Descriptions.....	1210
Table 12-13. RAMPHREFS Register Field Descriptions.....	1211
Table 12-14. RAMPHSTEPVALA Register Field Descriptions.....	1212
Table 12-15. RAMPHSTEPVALS Register Field Descriptions.....	1213
Table 12-16. RAMPHSTS Register Field Descriptions.....	1214
Table 12-17. DACLVALS Register Field Descriptions.....	1215
Table 12-18. DACLVALA Register Field Descriptions.....	1216
Table 12-19. RAMPHDLYA Register Field Descriptions.....	1217
Table 12-20. RAMPHDLYS Register Field Descriptions.....	1218
Table 12-21. CTRIPLFILCTL Register Field Descriptions.....	1219
Table 12-22. CTRIPLFILCLKCTL Register Field Descriptions.....	1220
Table 12-23. CTRIPHFILCTL Register Field Descriptions.....	1221
Table 12-24. CTRIPHFILCLKCTL Register Field Descriptions.....	1222
Table 12-25. COMPLOCK Register Field Descriptions.....	1223
Table 12-26. CMPSS_LITE_REGS Registers.....	1224
Table 12-27. CMPSS_LITE_REGS Access Type Codes.....	1224
Table 12-28. COMPCTL Register Field Descriptions.....	1225
Table 12-29. COMPHYCTL Register Field Descriptions.....	1227
Table 12-30. COMPSTS Register Field Descriptions.....	1228
Table 12-31. COMPSTCLR Register Field Descriptions.....	1229
Table 12-32. COMPDACHCTL Register Field Descriptions.....	1230
Table 12-33. DACHVALS Register Field Descriptions.....	1231
Table 12-34. DACHVALA Register Field Descriptions.....	1232
Table 12-35. DACLVALS Register Field Descriptions.....	1233
Table 12-36. DACLVALA Register Field Descriptions.....	1234
Table 12-37. CTRIPLFILCTL Register Field Descriptions.....	1235
Table 12-38. CTRIPLFILCLKCTL Register Field Descriptions.....	1236
Table 12-39. CTRIPHFILCTL Register Field Descriptions.....	1237
Table 12-40. CTRIPHFILCLKCTL Register Field Descriptions.....	1238
Table 12-41. COMPLOCK Register Field Descriptions.....	1239
Table 12-42. CMPSS Registers to Driverlib Functions.....	1239
Table 12-43. CMPSS_LITE Registers to Driverlib Functions.....	1241

Table 13-1. Submodule Configuration Parameters.....	1251
Table 13-2. Key Time-Base Signals.....	1255
Table 13-3. ePWM SYNC Selection.....	1260
Table 13-4. Action-Qualifier Submodule Possible Input Events.....	1275
Table 13-5. Action-Qualifier Event Priority for Up-Down-Count Mode.....	1277
Table 13-6. Action-Qualifier Event Priority for Up-Count Mode.....	1277
Table 13-7. Action-Qualifier Event Priority for Down-Count Mode.....	1277
Table 13-8. Behavior if CMPA/CMPB is Greater than the Period.....	1278
Table 13-9. Classical Dead-Band Operating Modes.....	1291
Table 13-10. Additional Dead-Band Operating Modes.....	1291
Table 13-11. Dead-Band Delay Values in $\mu$ s as a Function of DBFED and DBRED.....	1293
Table 13-12. Possible Pulse Width Values for EPWMCLK = 80MHz.....	1296
Table 13-13. Possible Actions On a Trip Event.....	1300
Table 13-14. Lock Bits and Corresponding Registers.....	1338
Table 13-15. Resolution for PWM and HRPWM.....	1340
Table 13-16. Relationship Between MEP Steps, PWM Frequency, and Resolution.....	1346
Table 13-17. CMPA versus Duty (left), and [CMPA:CMPAHR] versus Duty (right).....	1347
Table 13-18. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles.....	1350
Table 13-19. SFO Library Features.....	1362
Table 13-20. Factor Values.....	1363
Table 13-21. EPWM Base Address Table.....	1372
Table 13-22. EPWM_REGS Registers.....	1373
Table 13-23. EPWM_REGS Access Type Codes.....	1375
Table 13-24. TBCTL Register Field Descriptions.....	1376
Table 13-25. TBCTL2 Register Field Descriptions.....	1378
Table 13-26. EPWMSYNCSINSEL Register Field Descriptions.....	1379
Table 13-27. TBCTR Register Field Descriptions.....	1380
Table 13-28. TBSTS Register Field Descriptions.....	1381
Table 13-29. EPWMSYNCOUTEN Register Field Descriptions.....	1382
Table 13-30. TBCTL3 Register Field Descriptions.....	1384
Table 13-31. CMPCTL Register Field Descriptions.....	1385
Table 13-32. CMPCTL2 Register Field Descriptions.....	1387
Table 13-33. DBCTL Register Field Descriptions.....	1389
Table 13-34. DBCTL2 Register Field Descriptions.....	1392
Table 13-35. AQCTL Register Field Descriptions.....	1393
Table 13-36. AQTSRCSEL Register Field Descriptions.....	1395
Table 13-37. PCCTL Register Field Descriptions.....	1396
Table 13-38. VCAPCTL Register Field Descriptions.....	1398
Table 13-39. VCNTCFG Register Field Descriptions.....	1400
Table 13-40. HRCNFG Register Field Descriptions.....	1402
Table 13-41. HRPWR Register Field Descriptions.....	1404
Table 13-42. HRMSTEP Register Field Descriptions.....	1405
Table 13-43. HRCNFG2 Register Field Descriptions.....	1406
Table 13-44. HRPCTL Register Field Descriptions.....	1407
Table 13-45. TRREM Register Field Descriptions.....	1409
Table 13-46. GLDCTL Register Field Descriptions.....	1410
Table 13-47. GLDCFG Register Field Descriptions.....	1412
Table 13-48. EPWMXLINK Register Field Descriptions.....	1414
Table 13-49. AQCTLA Register Field Descriptions.....	1416
Table 13-50. AQCTLA2 Register Field Descriptions.....	1418
Table 13-51. AQCTLB Register Field Descriptions.....	1419
Table 13-52. AQCTLB2 Register Field Descriptions.....	1421
Table 13-53. AQSFRC Register Field Descriptions.....	1422
Table 13-54. AQCSFRC Register Field Descriptions.....	1423
Table 13-55. DBREDHR Register Field Descriptions.....	1424
Table 13-56. DBRED Register Field Descriptions.....	1425
Table 13-57. DBFEDHR Register Field Descriptions.....	1426
Table 13-58. DBFED Register Field Descriptions.....	1427
Table 13-59. TBPHS Register Field Descriptions.....	1428
Table 13-60. TBPRDHR Register Field Descriptions.....	1429
Table 13-61. TBPRD Register Field Descriptions.....	1430

Table 13-62. CMPA Register Field Descriptions.....	1431
Table 13-63. CMPB Register Field Descriptions.....	1432
Table 13-64. CMPC Register Field Descriptions.....	1433
Table 13-65. CMPD Register Field Descriptions.....	1434
Table 13-66. GLDCTL2 Register Field Descriptions.....	1435
Table 13-67. SWVDELVAL Register Field Descriptions.....	1436
Table 13-68. TZSEL Register Field Descriptions.....	1437
Table 13-69. TZDCSEL Register Field Descriptions.....	1439
Table 13-70. TZCTL Register Field Descriptions.....	1440
Table 13-71. TZCTL2 Register Field Descriptions.....	1442
Table 13-72. TZCTLDCA Register Field Descriptions.....	1444
Table 13-73. TZCTLDCA Register Field Descriptions.....	1446
Table 13-74. TZEINT Register Field Descriptions.....	1448
Table 13-75. TZFLG Register Field Descriptions.....	1449
Table 13-76. TZCBCFLG Register Field Descriptions.....	1451
Table 13-77. TZOSTFLG Register Field Descriptions.....	1453
Table 13-78. TZCLR Register Field Descriptions.....	1455
Table 13-79. TZCBCCLR Register Field Descriptions.....	1457
Table 13-80. TZOSTCLR Register Field Descriptions.....	1458
Table 13-81. TZFRC Register Field Descriptions.....	1459
Table 13-82. ETSEL Register Field Descriptions.....	1460
Table 13-83. ETPS Register Field Descriptions.....	1463
Table 13-84. ETFLG Register Field Descriptions.....	1466
Table 13-85. ETCLR Register Field Descriptions.....	1467
Table 13-86. ETFRC Register Field Descriptions.....	1468
Table 13-87. ETINTPS Register Field Descriptions.....	1469
Table 13-88. ETSOCPS Register Field Descriptions.....	1470
Table 13-89. ETCNTINITCTL Register Field Descriptions.....	1472
Table 13-90. ETCNTINIT Register Field Descriptions.....	1473
Table 13-91. DCTRIPSEL Register Field Descriptions.....	1474
Table 13-92. DCACTL Register Field Descriptions.....	1476
Table 13-93. DCBCTL Register Field Descriptions.....	1478
Table 13-94. DCFCTL Register Field Descriptions.....	1480
Table 13-95. DCCAPCTL Register Field Descriptions.....	1482
Table 13-96. DCFOFFSET Register Field Descriptions.....	1484
Table 13-97. DCFOFFSETCNT Register Field Descriptions.....	1485
Table 13-98. DCFWINDOW Register Field Descriptions.....	1486
Table 13-99. DCFWINDOWCNT Register Field Descriptions.....	1487
Table 13-100. BLANKPULSEMIXSEL Register Field Descriptions.....	1488
Table 13-101. DCCAP Register Field Descriptions.....	1490
Table 13-102. DCAHTRIPSEL Register Field Descriptions.....	1491
Table 13-103. DCALTRIPSEL Register Field Descriptions.....	1493
Table 13-104. DCBTRIPSEL Register Field Descriptions.....	1495
Table 13-105. DCBLTRIPSEL Register Field Descriptions.....	1497
Table 13-106. EPWMLOCK Register Field Descriptions.....	1499
Table 13-107. HWVDELVAL Register Field Descriptions.....	1501
Table 13-108. VCNTVAL Register Field Descriptions.....	1502
Table 13-109. EPWM Registers to Driverlib Functions.....	1503
Table 13-110. HRPWM Registers to Driverlib Functions.....	1510
Table 14-1. eCAP Input Selection.....	1518
Table 14-2. ECAP Base Address Table.....	1535
Table 14-3. ECAP_REGS Registers.....	1536
Table 14-4. ECAP_REGS Access Type Codes.....	1536
Table 14-5. TSCTR Register Field Descriptions.....	1537
Table 14-6. CTRPHS Register Field Descriptions.....	1538
Table 14-7. CAP1 Register Field Descriptions.....	1539
Table 14-8. CAP2 Register Field Descriptions.....	1540
Table 14-9. CAP3 Register Field Descriptions.....	1541
Table 14-10. CAP4 Register Field Descriptions.....	1542
Table 14-11. ECCTL0 Register Field Descriptions.....	1543
Table 14-12. ECCTL1 Register Field Descriptions.....	1544

Table 14-13. ECCTL2 Register Field Descriptions.....	1546
Table 14-14. ECEINT Register Field Descriptions.....	1548
Table 14-15. ECFLG Register Field Descriptions.....	1550
Table 14-16. ECCLR Register Field Descriptions.....	1552
Table 14-17. ECFRC Register Field Descriptions.....	1553
Table 14-18. ECAPSYNCINSEL Register Field Descriptions.....	1554
Table 14-19. ECAP Registers to Driverlib Functions.....	1554
Table 15-1. eQEP Input Source Select Table.....	1563
Table 15-2. EQEP Memory Map.....	1565
Table 15-3. Quadrature Decoder Truth Table.....	1567
Table 15-4. EQEP Base Address Table.....	1589
Table 15-5. EQEP_REGS Registers.....	1590
Table 15-6. EQEP_REGS Access Type Codes.....	1590
Table 15-7. QPOSCNT Register Field Descriptions.....	1592
Table 15-8. QPOSINIT Register Field Descriptions.....	1593
Table 15-9. QPOSMAX Register Field Descriptions.....	1594
Table 15-10. QPOSCMP Register Field Descriptions.....	1595
Table 15-11. QPOSILAT Register Field Descriptions.....	1596
Table 15-12. QPOSSLAT Register Field Descriptions.....	1597
Table 15-13. QPOSLAT Register Field Descriptions.....	1598
Table 15-14. QUTMR Register Field Descriptions.....	1599
Table 15-15. QUPRD Register Field Descriptions.....	1600
Table 15-16. QWDTMR Register Field Descriptions.....	1601
Table 15-17. QWDPRD Register Field Descriptions.....	1602
Table 15-18. QDECCTL Register Field Descriptions.....	1603
Table 15-19. QEPCTL Register Field Descriptions.....	1605
Table 15-20. QCAPCTL Register Field Descriptions.....	1607
Table 15-21. QPOSCTL Register Field Descriptions.....	1608
Table 15-22. QEINT Register Field Descriptions.....	1609
Table 15-23. QFLG Register Field Descriptions.....	1611
Table 15-24. QCLR Register Field Descriptions.....	1613
Table 15-25. QFRC Register Field Descriptions.....	1615
Table 15-26. QEPSTS Register Field Descriptions.....	1617
Table 15-27. QCTMR Register Field Descriptions.....	1619
Table 15-28. QCPRD Register Field Descriptions.....	1620
Table 15-29. QCTMRLAT Register Field Descriptions.....	1621
Table 15-30. QCPRDLAT Register Field Descriptions.....	1622
Table 15-31. REV Register Field Descriptions.....	1623
Table 15-32. QEPSTROBESEL Register Field Descriptions.....	1624
Table 15-33. QMACTRL Register Field Descriptions.....	1625
Table 15-34. QEPSRCSEL Register Field Descriptions.....	1626
Table 15-35. EQEP Registers to Driverlib Functions.....	1627
Table 16-1. CAN Register Access from Software.....	1636
Table 16-2. CAN Register Access from Code Composer Studio™ IDE.....	1636
Table 16-3. PIE Module Nomenclature for Interrupts.....	1644
Table 16-4. Programmable Ranges Required by CAN Protocol.....	1656
Table 16-5. Message Object Field Descriptions.....	1666
Table 16-6. Message RAM Addressing in Debug Mode.....	1669
Table 16-7. CAN Base Address Table.....	1673
Table 16-8. CAN_REGS Registers.....	1674
Table 16-9. CAN_REGS Access Type Codes.....	1675
Table 16-10. CAN_CTL Register Field Descriptions.....	1676
Table 16-11. CAN_ES Register Field Descriptions.....	1679
Table 16-12. CAN_ERRC Register Field Descriptions.....	1681
Table 16-13. CAN_BTR Register Field Descriptions.....	1682
Table 16-14. CAN_INT Register Field Descriptions.....	1684
Table 16-15. CAN_TEST Register Field Descriptions.....	1685
Table 16-16. CAN_PERR Register Field Descriptions.....	1687
Table 16-17. CAN_RAM_INIT Register Field Descriptions.....	1688
Table 16-18. CAN_GLB_INT_EN Register Field Descriptions.....	1689
Table 16-19. CAN_GLB_INT_FLG Register Field Descriptions.....	1690

Table 16-20. CAN_GLB_INT_CLR Register Field Descriptions.....	1691
Table 16-21. CAN_ABOTR Register Field Descriptions.....	1692
Table 16-22. CAN_TXRQ_X Register Field Descriptions.....	1693
Table 16-23. CAN_TXRQ_21 Register Field Descriptions.....	1694
Table 16-24. CAN_NDAT_X Register Field Descriptions.....	1695
Table 16-25. CAN_NDAT_21 Register Field Descriptions.....	1696
Table 16-26. CAN_IPEN_X Register Field Descriptions.....	1697
Table 16-27. CAN_IPEN_21 Register Field Descriptions.....	1698
Table 16-28. CAN_MVAL_X Register Field Descriptions.....	1699
Table 16-29. CAN_MVAL_21 Register Field Descriptions.....	1700
Table 16-30. CAN_IP_MUX21 Register Field Descriptions.....	1701
Table 16-31. CAN_IF1CMD Register Field Descriptions.....	1702
Table 16-32. CAN_IF1MSK Register Field Descriptions.....	1705
Table 16-33. CAN_IF1ARB Register Field Descriptions.....	1706
Table 16-34. CAN_IF1MCTL Register Field Descriptions.....	1708
Table 16-35. CAN_IF1DATA Register Field Descriptions.....	1710
Table 16-36. CAN_IF1DATB Register Field Descriptions.....	1711
Table 16-37. CAN_IF2CMD Register Field Descriptions.....	1712
Table 16-38. CAN_IF2MSK Register Field Descriptions.....	1715
Table 16-39. CAN_IF2ARB Register Field Descriptions.....	1716
Table 16-40. CAN_IF2MCTL Register Field Descriptions.....	1718
Table 16-41. CAN_IF2DATA Register Field Descriptions.....	1720
Table 16-42. CAN_IF2DATB Register Field Descriptions.....	1721
Table 16-43. CAN_IF3MSK Register Field Descriptions.....	1722
Table 16-44. CAN_IF3ARB Register Field Descriptions.....	1723
Table 16-45. CAN_IF3MCTL Register Field Descriptions.....	1724
Table 16-46. CAN_IF3DATA Register Field Descriptions.....	1726
Table 16-47. CAN_IF3DATB Register Field Descriptions.....	1727
Table 16-48. CAN_IF3UPD Register Field Descriptions.....	1728
Table 16-49. CAN Registers to Driverlib Functions.....	1728
Table 17-1. Dependency of Delay d on the Divide-Down Value IPSC.....	1738
Table 17-2. Operating Modes of the I2C Module.....	1740
Table 17-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMR.....	1740
Table 17-4. How the MST and FDF Bits of I2CMR Affect the Role of the TRX Bit of I2CMR.....	1746
Table 17-5. Ways to Generate a NACK Bit.....	1750
Table 17-6. Descriptions of the Basic I2C Interrupt Requests.....	1751
Table 17-7. I2C Base Address Table.....	1757
Table 17-8. I2C_REGS Registers.....	1758
Table 17-9. I2C_REGS Access Type Codes.....	1758
Table 17-10. I2COAR Register Field Descriptions.....	1759
Table 17-11. I2CIER Register Field Descriptions.....	1760
Table 17-12. I2CSTR Register Field Descriptions.....	1761
Table 17-13. I2CCLKL Register Field Descriptions.....	1765
Table 17-14. I2CCLKH Register Field Descriptions.....	1766
Table 17-15. I2CCNT Register Field Descriptions.....	1767
Table 17-16. I2CDRR Register Field Descriptions.....	1768
Table 17-17. I2CSAR Register Field Descriptions.....	1769
Table 17-18. I2CDXR Register Field Descriptions.....	1770
Table 17-19. I2CMR Register Field Descriptions.....	1771
Table 17-20. I2CISRC Register Field Descriptions.....	1775
Table 17-21. I2CEMR Register Field Descriptions.....	1776
Table 17-22. I2CPSC Register Field Descriptions.....	1777
Table 17-23. I2CFFTX Register Field Descriptions.....	1778
Table 17-24. I2CFFRX Register Field Descriptions.....	1780
Table 17-25. I2C Registers to Driverlib Functions.....	1781
Table 18-1. SCI Module Signal Summary.....	1785
Table 18-2. Programming the Data Format Using SCICCR.....	1788
Table 18-3. Asynchronous Baud Register Values for Common SCI Bit Rates.....	1798
Table 18-4. SCI Interrupt Flags.....	1800
Table 18-5. SCI Base Address Table.....	1804
Table 18-6. SCI_REGS Registers.....	1805

Table 18-7. SCI_REGS Access Type Codes.....	1805
Table 18-8. SCICCR Register Field Descriptions.....	1806
Table 18-9. SCICTL1 Register Field Descriptions.....	1808
Table 18-10. SCIHBAUD Register Field Descriptions.....	1810
Table 18-11. SCILBAUD Register Field Descriptions.....	1811
Table 18-12. SCICTL2 Register Field Descriptions.....	1812
Table 18-13. SCIRXST Register Field Descriptions.....	1814
Table 18-14. SCIRXEMU Register Field Descriptions.....	1818
Table 18-15. SCIRXBUF Register Field Descriptions.....	1819
Table 18-16. SCITXBUF Register Field Descriptions.....	1821
Table 18-17. SCIFFTX Register Field Descriptions.....	1822
Table 18-18. SCIFFRX Register Field Descriptions.....	1824
Table 18-19. SCIFFCT Register Field Descriptions.....	1826
Table 18-20. SCIPRI Register Field Descriptions.....	1827
Table 18-21. SCI Registers to Driverlib Functions.....	1827
Table 19-1. SPI Module Signal Summary.....	1834
Table 19-2. SPI Interrupt Flag Modes.....	1836
Table 19-3. SPI Clocking Scheme Selection Guide.....	1843
Table 19-4. 4-wire versus 3-wire SPI Pin Functions.....	1845
Table 19-5. 3-Wire SPI Pin Configuration.....	1846
Table 19-6. SPI Base Address Table.....	1854
Table 19-7. SPI_REGS Registers.....	1855
Table 19-8. SPI_REGS Access Type Codes.....	1855
Table 19-9. SPICCR Register Field Descriptions.....	1856
Table 19-10. SPICTL Register Field Descriptions.....	1858
Table 19-11. SPISTS Register Field Descriptions.....	1860
Table 19-12. SPIBRR Register Field Descriptions.....	1862
Table 19-13. SPIRXEMU Register Field Descriptions.....	1863
Table 19-14. SPIRXBUF Register Field Descriptions.....	1864
Table 19-15. SPITXBUF Register Field Descriptions.....	1865
Table 19-16. SPIDAT Register Field Descriptions.....	1866
Table 19-17. SPIFFTX Register Field Descriptions.....	1867
Table 19-18. SPIFFRX Register Field Descriptions.....	1869
Table 19-19. SPIFFCT Register Field Descriptions.....	1871
Table 19-20. SPIPRI Register Field Descriptions.....	1872
Table 19-21. SPI Registers to Driverlib Functions.....	1873
Table 20-1. SIGGENx Active Register Loading.....	1880
Table 20-2. EPG Data Input Connections.....	1883
Table 20-3. EPG Input Connections.....	1884
Table 20-4. EPG Output Connections.....	1884
Table 20-5. EPG Base Address Table.....	1890
Table 20-6. EPG_REGS Registers.....	1891
Table 20-7. EPG_REGS Access Type Codes.....	1891
Table 20-8. GCTL0 Register Field Descriptions.....	1893
Table 20-9. GCTL1 Register Field Descriptions.....	1895
Table 20-10. GCTL2 Register Field Descriptions.....	1896
Table 20-11. GCTL3 Register Field Descriptions.....	1898
Table 20-12. EPGLOCK Register Field Descriptions.....	1902
Table 20-13. EPGCOMMIT Register Field Descriptions.....	1903
Table 20-14. GINTSTS Register Field Descriptions.....	1904
Table 20-15. GINTEN Register Field Descriptions.....	1905
Table 20-16. GINTCLR Register Field Descriptions.....	1906
Table 20-17. GINTFRC Register Field Descriptions.....	1907
Table 20-18. CLKDIV0_CTL0 Register Field Descriptions.....	1908
Table 20-19. CLKDIV0_CLKOFFSET Register Field Descriptions.....	1909
Table 20-20. CLKDIV1_CTL0 Register Field Descriptions.....	1910
Table 20-21. CLKDIV1_CLKOFFSET Register Field Descriptions.....	1911
Table 20-22. SIGGEN0_CTL0 Register Field Descriptions.....	1912
Table 20-23. SIGGEN0_CTL1 Register Field Descriptions.....	1914
Table 20-24. SIGGEN0_DATA0 Register Field Descriptions.....	1915
Table 20-25. SIGGEN0_DATA1 Register Field Descriptions.....	1916

---

Table 20-26. SIGGEN0_DATA0_ACTIVE Register Field Descriptions.....	1917
Table 20-27. SIGGEN0_DATA1_ACTIVE Register Field Descriptions.....	1918
Table 20-28. EPG_MUX_REGS Registers.....	1919
Table 20-29. EPG_MUX_REGS Access Type Codes.....	1919
Table 20-30. EPGMXSEL0 Register Field Descriptions.....	1920
Table 20-31. EPGMXSELLOCK Register Field Descriptions.....	1923
Table 20-32. EPGMXSELCOMMIT Register Field Descriptions.....	1924
Table 20-33. EPG Registers to Driverlib Functions.....	1924

This page intentionally left blank.



## About This Manual

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

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

## Notational Conventions

This document uses the following conventions.

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

## Glossary

[TI Glossary](#) This glossary lists and explains terms, acronyms, and definitions.

## Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for these devices, visit the Texas Instruments website at [www.ti.com](http://www.ti.com).

Additionally, the [TMS320C28x DSP CPU and Instruction Set Reference Guide](#) and the [TMS320C28x Floating Point Unit and Instruction Set Reference Guide](#) must be used in conjunction with this TRM.

## Support Resources

[TI E2E™ support forums](#) are an engineer's go-to source for fast, verified answers and design help — straight from the experts. Search existing answers or ask your own question to get the quick design help you need.

Linked content is provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's [Terms of Use](#).

**Trademarks**

TI E2E™, C2000™, Code Composer Studio™, and Texas Instruments™ are trademarks of Texas Instruments. All trademarks are the property of their respective owners.

Chapter 1

# C2000™ Microcontrollers Software Support

---



This chapter discusses the C2000Ware for the C2000™ microcontrollers. The C2000Ware can be downloaded from: [www.ti.com/tool/C2000WARE](http://www.ti.com/tool/C2000WARE)

<b>1.1 Introduction</b> .....	<b>54</b>
<b>1.2 C2000Ware Structure</b> .....	<b>54</b>
<b>1.3 Documentation</b> .....	<b>54</b>
<b>1.4 Devices</b> .....	<b>54</b>
<b>1.5 Libraries</b> .....	<b>54</b>
<b>1.6 Code Composer Studio™ Integrated Development Environment (IDE)</b> .....	<b>54</b>
<b>1.7 SysConfig and PinMUX Tool</b> .....	<b>55</b>

## 1.1 Introduction

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

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

## 1.2 C2000Ware Structure

The C2000Ware software package is organized into the following directory structure as shown in [Table 1-1](#).

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

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

## 1.3 Documentation

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

## 1.4 Devices

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

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

## 1.5 Libraries

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

## 1.6 Code Composer Studio™ Integrated Development Environment (IDE)

Code Composer Studio™ is an integrated development environment (IDE) that supports TI's microcontroller and embedded processors portfolio. The Code Composer Studio™ IDE comprises a suite of tools used to develop and debug embedded applications. The latest version of Code Composer Studio™ IDE can be obtained at: [www.ti.com/ccstudio](http://www.ti.com/ccstudio)

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

## 1.7 SysConfig and PinMUX Tool

To help simplify configuration challenges and accelerate software development, Texas Instruments™ created SysConfig, an intuitive and comprehensive collection of graphical utilities for configuring pins, peripherals, subsystems, and other components. SysConfig helps you manage, expose, and resolve conflicts visually so that you have more time to create differentiated applications.

The tool's output includes C header and code files that can be used with C2000Ware examples or used to configure custom software.

The SysConfig tool automatically selects the pinmux settings that satisfy the entered requirements. The SysConfig tool is delivered integrated in the Code Composer Studio™ IDE, in the C2000Ware GPIO example, as a standalone installer, or can be used by way of the cloud tools portal at: [dev.ti.com](https://dev.ti.com)

This page intentionally left blank.



This chapter contains a short description of the C28x processor and extended instruction sets.

Further information can be found in the following documents:

- [TMS320C28x CPU and Instruction Set Reference Guide](#)
- [TMS320C28x Extended Instruction Sets Technical Reference Manual](#)
- [Accelerators: Enhancing the Capabilities of the C2000 MCU Family Technical Brief](#)
- [TMS320C28x FPU Primer Application Report](#)

<b>2.1 Introduction</b> .....	<b>58</b>
<b>2.2 C28X Related Collateral</b> .....	<b>58</b>
<b>2.3 Features</b> .....	<b>58</b>
<b>2.4 Floating-Point Unit</b> .....	<b>59</b>
<b>2.5 Trigonometric Math Unit (TMU)</b> .....	<b>59</b>

## 2.1 Introduction

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

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

## 2.2 C28X Related Collateral

### Foundational Materials

- [C2000 Academy - C28x](#)
- [C2000 C28x Migration from COFF to EABI](#)
- [C2000 C28x Optimization Guide](#)
- [C2000 Performance Tips and Tricks](#)
- [C2000 Software Guide](#)
- [CGT Data Blocking C2000](#)
- [Enhancing the Computational Performance of the C2000™ Microcontroller Family Application Report](#)

### Getting Started Materials

- [C2000 Multicore Development User Guide](#)
- [C2000Ware - CLAMath](#)
- [C2000Ware - FPU Fast RTS](#)
- [C2000Ware - FPU Library](#)
- [C2000Ware - Fast Integer Division](#)
- [C2000Ware - Fixed Point Library](#)
- [C2000Ware - IQMath](#)
- [C2000Ware - VCU Library](#)
- [C28x Context Save and Restore](#)
- [CRC Engines in C2000 Devices Application Report](#)
- [TMS320C28x Extended Instruction Sets Application Report](#)
- [TMS320C28x FPU Primer Application Report](#)

## 2.3 Features

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

## 2.4 Floating-Point Unit

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

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

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

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

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

## 2.5 Trigonometric Math Unit (TMU)

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

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

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

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

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

This page intentionally left blank.

## Chapter 3 System Control and Interrupts



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

<b>3.1 Introduction</b> .....	62
<b>3.2 Power Management</b> .....	63
<b>3.3 Device Identification and Configuration Registers</b> .....	63
<b>3.4 Resets</b> .....	63
<b>3.5 Peripheral Interrupts</b> .....	66
<b>3.6 Exceptions and Non-Maskable Interrupts</b> .....	77
<b>3.7 Clocking</b> .....	78
<b>3.8 32-Bit CPU Timers 0/1/2</b> .....	94
<b>3.9 Watchdog Timer</b> .....	95
<b>3.10 Low-Power Modes</b> .....	98
<b>3.11 Memory Controller Module</b> .....	101
<b>3.12 JTAG</b> .....	105
<b>3.13 System Control Register Configuration Restrictions</b> .....	106
<b>3.14 Software</b> .....	107
<b>3.15 System Control Registers</b> .....	114

## 3.1 Introduction

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

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

### 3.1.1 SYSTCTL Related Collateral

#### Foundational Materials

- [C2000 MCU JTAG Connectivity Debug Application Report](#)

#### Getting Started Materials

- [C28x Interrupt Nesting](#)
- [Debugging JTAG](#)
- [Enhancing Device Security by Using JTAGLOCK Feature Application Report](#)
- [Interrupt FAQ for C2000](#)
- [XDS Target Connection Guide](#)

#### Expert Materials

- [C2000 CPU Memory Built-In Self-Test Application Report](#)
- [C2000 Memory Power-On Self-Test \(M-POST\) Application Report](#)
- [Programming of External Nonvolatile Memory Using SDFlash for TMS320C28x Devices Application Report](#)
- [Software Phased-Locked Loop \(PLL\) Design Using C2000 Microcontrollers Application Report](#)

### 3.1.2 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 the register descriptions for details.

### 3.1.3 EALLOW Protection

Some registers in the system are protected from spurious CPU writes by the EALLOW protection mechanism. This uses the special CPU instructions EALLOW and EDIS to enable and disable access to protected registers. The current protection state is given by the EALLOW bit in the CPU ST1 register, as shown in [Table 3-1](#).

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

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

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

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

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

## 3.2 Power Management

The TMS320F280013x MCU 1.2V core is powered up internally with the LDO (VREG). VREG power is derived from the 3.3V rail so this device can support a single 3.3V rail operation.

## 3.3 Device Identification and Configuration Registers

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

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

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

- UID\_PSRAND0-4: 160 bits of pseudo-random data
- UID\_UNIQUE: 64-bit unique data; the value in this register is unique across all devices in the same PARTIDH
- UID\_CHECKSUM: 32-bit Fletcher checksum of UID\_PSRAND0-4 and UID\_UNIQUE and calculated as either little- or big-endian during factory testing

## 3.4 Resets

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

### 3.4.1 Reset Sources

Table 3-2 summarizes the various reset signals and the effect on the device.

**Table 3-2. Reset Signals**

Reset Source	CPU Core Reset (C28x, FPU, VCU)	Peripherals Reset	JTAG / Debug Logic Reset	IOs	XRS Output
POR	Yes	Yes	Yes	Hi-Z	Yes
BOR	Yes	Yes	Yes	Hi-Z	Yes
XRS Pin	Yes	Yes	No	Hi-Z	-
$\overline{WDRS}$	Yes	Yes	No	Hi-Z	Yes
$\overline{NMIWDRS}$	Yes	Yes	No	Hi-Z	Yes
$\overline{SYSRS}$ (Debugger Reset)	Yes	Yes	No	Hi-Z	No
$\overline{SCCRESET}$	Yes	Yes	No	Hi-Z	No
SIMRESET. $\overline{XRS}$	Yes	Yes	No	Hi-Z	Yes
SIMRESET. $\overline{CPU1RS}$	Yes	Yes	No	Hi-Z	No

The resets can be divided into two groups:

- Chip-level resets ( $\overline{XRS}$ , POR, BOR,  $\overline{WDRS}$ , SIMRESET. $\overline{XRS}$  and  $\overline{NMIWDRS}$ ), which reset all or almost all of the device.
- System resets ( $\overline{SYSRS}$ , SIMRESET. $\overline{CPU1RS}$  and  $\overline{SCCRESET}$ ), which reset a large subset of the device but maintain some system-level configuration.

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

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

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

---

#### Note

After a POR, the boot ROMs clear the M0/M1 and LSx RAMs to make sure that the RAMs contain valid ECC or parity.

---

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

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

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

### 3.4.3 Simulate External Reset (SIMRESET. $\overline{XRS}$ )

The user can simulate an external reset ( $\overline{XRS}$ ) in software. This can be done by setting the XRSn bit to 1 in the SIMRESET register by software. This toggles the  $\overline{XRS}$  pin; hence, resetting the full device (just like external reset).

After this reset, the SIMRESET\_XRSn and XRSn bits in the RESC register are set. Software can read these bits to know the cause of the reset and clear the status by writing a 1 into the corresponding bits in the RESCCLR register.

### 3.4.4 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 can require a longer pulse. In these cases, the  $\overline{XRS}$  pin can be driven low externally to provide the correct reset duration. A POR resets everything that  $\overline{XRS}$  does, along with a few other registers – the reset cause register (RESC), the NMI shadow flag register (NMISHDFLG), and the X1 clock counter register (X1CNT). A POR also resets the debug logic used by the JTAG port.

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

### 3.4.5 Brown-Out-Reset (BOR)

The brown-out-reset (BOR) is an internal supply voltage supervisor (SVS) circuit which monitors the VDDIO supply for glitches or supply interruptions. If the VDDIO supply voltage drops below operational voltage range, this circuit forces the XRSn pin low until the fault is removed and the supply voltage returns to the minimum operational voltage. A BOR resets everything in the same manner as a POR reset.

The BOR circuit is enabled by default and therefore is always active during power up or after any type of reset. To disable the BOR circuit, set the BORLVMONDIS bit in the VMONCTL register.

### 3.4.6 Debugger Reset ( $\overline{\text{SYSRS}}$ )

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

The  $\overline{\text{SYSRS}}$  does not reset the ICEPick debug module, the device capability registers, the clock source and PLL configurations, the missing clock detection state, the PIE vector fetch error handler address, the NMI flags, the analog trims, or anything reset only by a POR (see [Section 3.4.4](#)).

### 3.4.7 Simulate CPU Reset

The user can simulate a CPU reset ( $\overline{\text{SYSRS}}$ ) in software. This can be done by setting CPU1RSn bit to 1 in the SIMRESET register by CPU1 software. This toggles the CPU1.SYSRS signals; hence, resetting the CPU (just like the debugger reset).

After this reset, the SIMRESET\_CPU1RSn bit in the RESC register is set. Software can read this bit to know the cause of the reset and clear the status by writing a 1 into the corresponding bit in the RESCCLR register.

### 3.4.8 Watchdog Reset ( $\overline{\text{WDRS}}$ )

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

After a watchdog reset, the WDRSn and XRSn bits in RESC are set.

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

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

After an NMI watchdog reset, the NMIWDRSn and XRSn bits in RESC are set.

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

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

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

## 3.5 Peripheral Interrupts

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

### 3.5.1 Interrupt Concepts

An interrupt is a signal that causes the CPU to pause the 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.

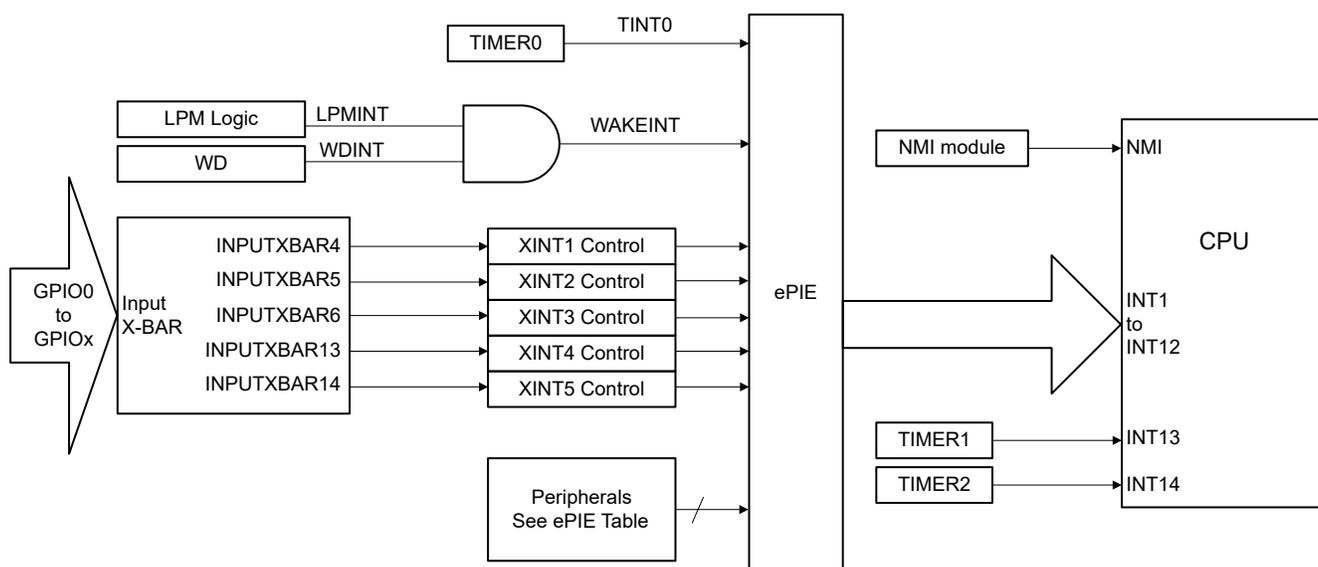
Interrupts propagate to the CPU through a series of flag and enable registers. The flag registers store the interrupt until the interrupt is processed. The enable registers block the propagation of the interrupt. When an interrupt signal reaches the CPU, the CPU fetches the appropriate ISR address from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

### 3.5.2 Interrupt Architecture

The C28x CPU has 14 peripheral interrupt lines. Two of them (INT13 and INT14) are connected directly to CPU timers 1 and 2, respectively. The remaining 12 interrupt lines 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 eight peripheral interrupts into each CPU interrupt line. The PIE also expands the vector table to allow each interrupt to have an 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 enable and flag registers. This system allows the CPU to handle one interrupt while others are pending, implement and prioritize nested interrupts in software, and disable interrupts during certain critical tasks.

[Figure 3-1](#) shows the interrupt architecture for this device.



**Figure 3-1. Device Interrupt Architecture**

### 3.5.2.1 Peripheral Stage

Each peripheral has a 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 can 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 is generated.

### 3.5.2.2 PIE Stage

The PIE provides individual flag and enable register bits for each of the peripheral interrupt signals, which are sometimes called PIE channels. These channels are grouped according to the associated CPU interrupt. Each PIE group has one 8-bit enable register (PIEIERx), one 8-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, the CPU 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 does not happen unless software changes the state of the PIE while an interrupt is propagating. Section 3.5.4 contains procedures for safely modifying the PIE configuration once interrupts have been enabled.

### 3.5.2.3 CPU Stage

Like the PIE, the CPU provides flag and enable register bits for each of the 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 SETC and CLRC instructions. In C code, C2000Ware's DINT and EINT macros can be used for this purpose.

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

### 3.5.3 Interrupt Entry Sequence

Figure 3-2 shows how peripheral interrupts propagate to the CPU.

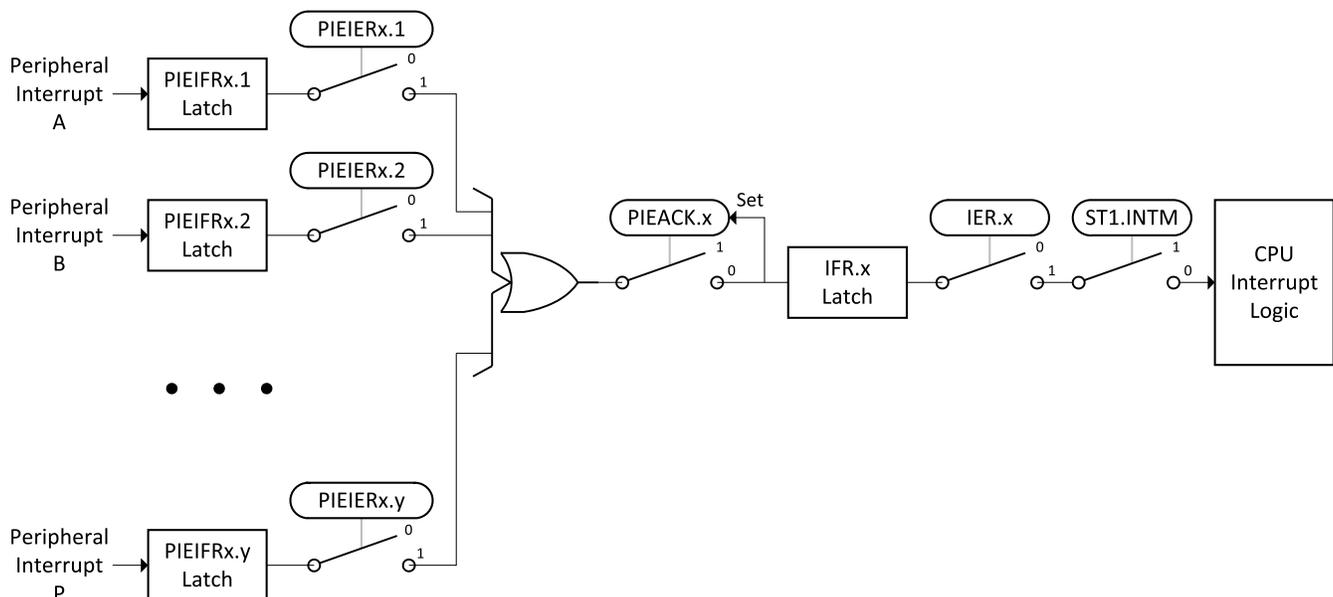


Figure 3-2. Interrupt Propagation Path

When a peripheral generates an interrupt (on PIE group x, channel y), the interrupt 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 the 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 add to the latency. External interrupts add a minimum of 2 SYSCLK cycles for GPIO synchronization plus extra time for input qualification (if used). Loops created using the C28x RPT instruction cannot be interrupted.

### 3.5.4 Configuring and Using Interrupts

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

#### 3.5.4.1 Enabling Interrupts

To enable a peripheral interrupt, perform the following steps:

1. Disable interrupts globally (DINT or SETC INTM).
2. Enable the PIE by setting the ENPIE bit of the PIECTRL register.
3. Write the ISR vector for each interrupt to the appropriate location in the PIE vector table, which can be found in [Section 3.5.8](#). Note that the vector table is EALLOW-protected.
4. Set the appropriate PIEIERx bit for each interrupt. The PIE group and channel assignments can be found in [Section 3.5.8](#).
5. Set the CPU IER bit for any PIE group containing enabled interrupts.
6. Enable the interrupt in the peripheral.
7. Enable interrupts globally (EINT or CLRC INTM).

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

#### 3.5.4.2 Handling Interrupts

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

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

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

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

### 3.5.4.3 Disabling Interrupts

To disable all interrupts, set the CPU global interrupt mask using DINT or SETC INTM. It is not necessary to add NOPs after setting INTM or modifying IER – the next instruction executes 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, the signal can 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 can 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 interrupt vector to an empty ISR. This ISR only contains a return from interrupt (IRET) instruction.
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 the original ISR.
8. Clear the PIEACK bit for the interrupt's PIE group.
9. Enable interrupts globally.

### 3.5.4.4 Nesting Interrupts

By default, interrupts do not nest. It is possible to nest and prioritize interrupts using software control of the IER and PIEIERx registers. Example code can be found in C2000Ware and documentation is available at [software-dl.ti.com/C2000/docs/c28x\\_interrupt\\_nesting/html/index.html](https://software-dl.ti.com/C2000/docs/c28x_interrupt_nesting/html/index.html).

### 3.5.4.5 Vector Address Validity Check

The ePIE vector table memory is protected using a parity check. Upon each vector fetch from the ePIE, a parity check is performed. If a parity failure occurs during vector fetch, the ePIE returns either a user defined error handler routine (if PIEVERRADDR is defined with a non 0x003F FFFF value), or the default boot ROM handler at address 0x3F FFBE. The ePIE also sends trip signals to the EPWMs.

The parity check only returns the error handler value if the failure occurs during vector fetch. Parity errors during a data read are handled by the memory controller module and logged by the UCERRFLG register in the MEMORY\_ERROR\_REGS. The address that caused the error is located in the UCCPUREADDR register. If the error address logged is between 0xD00 to 0xDFF, then the error is a PIE parity error. Additionally, a parity error during a vector fetch does not flag an uncorrectable error NMI.

### 3.5.5 PIE Channel Mapping

Table 3-3 shows the PIE group and channel assignments for each peripheral interrupt. Each row is a group, and each column is a channel within that group. When multiple interrupts are pending, the lowest-numbered channel in 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.

#### Note

Cells marked "-" are Reserved.

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

	INTx.1	INTx.2	INTx.3	INTx.4	INTx.5	INTx.6	INTx.7	INTx.8
INT1.y	ADCA1	ADCC1	-	XINT1	XINT2	SYS_ERR	TIMER0	WAKE
INT2.y	EPWM1_TZ	EPWM2_TZ	EPWM3_TZ	EPWM4_TZ	EPWM5_TZ	EPWM6_TZ	EPWM7_TZ	-
INT3.y	EPWM1	EPWM2	EPWM3	EPWM4	EPWM5	EPWM6	EPWM7	-
INT4.y	ECAP1	ECAP2	-	-	-	-	-	-
INT5.y	EQEP1	-	-	-	-	-	-	-
INT6.y	SPIA_RX	SPIA_TX	-	-	-	-	DCC0	-
INT7.y	-	-	-	-	-	-	-	-
INT8.y	I2CA	I2CA_FIFO	I2CB	I2CB_FIFO	SCIC_RX	SCIC_TX	-	-
INT9.y	SCIA_RX	SCIA_TX	SCIB_RX	SCIB_TX	CANA_0	CANA_1	-	-
INT10.y	ADCA_EVT	ADCA2	ADCA3	ADCA4	ADCC_EVT	ADCC2	ADCC3	ADCC4
INT11.y	-	-	-	-	-	-	-	-
INT12.y	XINT3	XINT4	XINT5	-	FLSS_INT	-	-	-

### 3.5.6 PIE Interrupt Priority

#### 3.5.6.1 Channel Priority

For every PIE group, the low-number channels in the group have the highest priority. For instance in PIE group 1, channel 1.1 has priority over channel 1.3. If those two enabled interrupts occurred simultaneously, channel 1.1 is serviced first with channel 1.3 left pending. Once the ISR for channel 1.1 completes and provided there are no other enabled and pending interrupts for PIE group 1, channel 1.3 is serviced. However, for the CPU to service any more interrupts from a PIE group, PIEACK for the group must be cleared. For this specific example, for channel 1.3 to be serviced, the channel 1.1 ISR has to clear PIEACK for group 1.

The following example describes an alternative scenario: channel 1.1 is currently being serviced by the CPU, channel 1.3 is pending and before the channel 1.1 ISR completes, channel 1.2 that is enabled also comes in. Since channel 1.2 has a higher priority than channel 1.3, the CPU services channel 1.2 and channel 1.3 is still left pending. Using the steps from the Interrupt Entry Sequence ([Section 3.5.3](#)), a channel 1.2 interrupt can happen as late as step 10 (The CPU fetches the ISR vector from the PIE. PIEIFRx.y is cleared) and the interrupt is still serviced ahead of channel 1.3.

#### 3.5.6.2 Group Priority

Generally, the lowest channel in the lowest PIE group has the highest priority. An example of this is channels 1.1 and 2.1. Those two channels have the highest priority in the respective groups. If the interrupts for those two enabled channels happened simultaneously and provided there are no other enabled and pending interrupts, channel 1.1 is serviced first by the CPU with channel 2.1 left pending.

However, there are cases where channel priority supersedes group priority. This special case happens depending on which step the CPU is currently at in the Interrupt Entry Sequence ([Section 3.5.3](#)).

The following illustrates an example of this special case.

The CPU is about to service channel 2.3 and is currently going through the steps in the Interrupt Entry Sequence ([Section 3.5.3](#)).

1. As the CPU reaches step 10 (The CPU fetches the ISR vector from the PIE. PIEIFRx.y is cleared), two enabled interrupts: channel 1.1 and channel 2.1 come in.
2. Due to channel priority, channel 2.1 is serviced ahead of channel 2.3. However, group priority dictates that channel 1.1 be serviced ahead of channels 2.1 and 2.3.
3. Channel priority supersedes here and channel 2.1 is serviced ahead of channels 1.1 and 2.3.
4. After channel 2.1 completes, channel 1.1 is serviced followed by channel 2.3.

Group priority is only maintained if no interrupts are currently being serviced, that is, the Interrupt Entry Sequence ([Section 3.5.3](#)) is not executing.

### 3.5.7 System Error

SYS\_ERR consolidate several sources of interrupts (see Figure 3-3). These sources set the respective bit in the SYS\_ERR\_INT\_FLG register. Any set bit in the SYS\_ERR\_INT\_FLG register also sets the global interrupt (GINT) bit. The GINT bit has to be cleared before any SYS\_ERR interrupt is generated. If the GINT bit is cleared with the source flags still set, another SYS\_ERR interrupt is fired; therefore, it is recommended to clear the source flags before clearing the GINT bit.

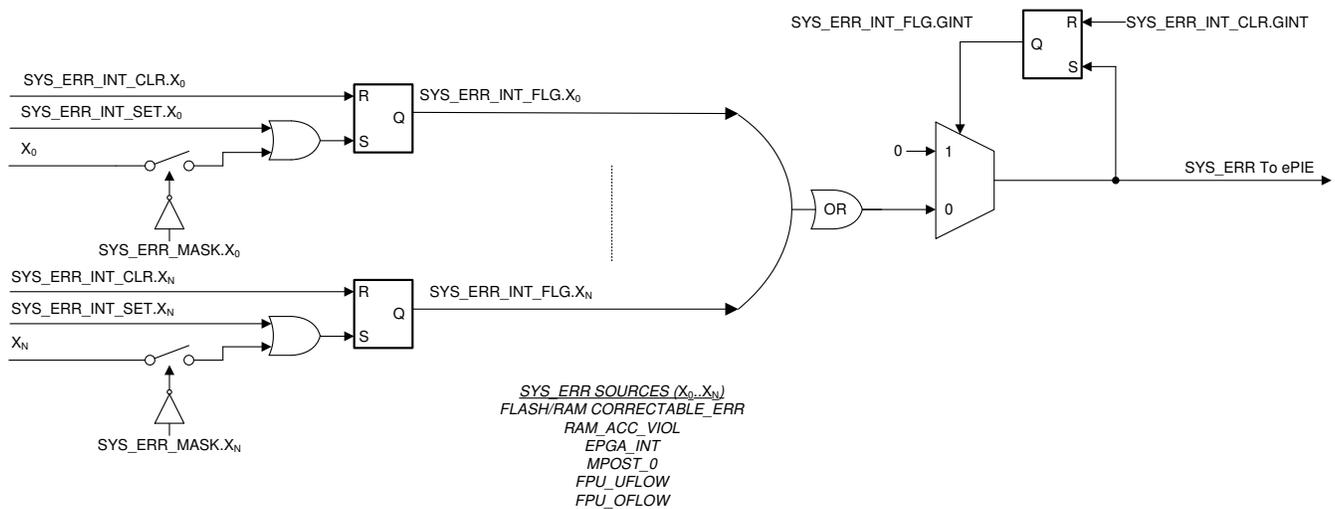


Figure 3-3. System Error

### 3.5.8 Vector Tables

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

Table 3-5 shows the PIE vector table.

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

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

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

Name	Vector ID	Address	Size (x16)	Description	Core Priority	ePIE Group priority
<b>PIE Group 1 Vectors - Muxed into CPU INT1</b>						
INT1.1	32	0x0000 0D40	2	ADCA1 interrupt	5	1 (Highest)
INT1.2	33	0x0000 0D42	2	ADCC1 interrupt	5	2
INT1.3	34	0x0000 0D44	2	Reserved	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	SYS_ERR interrupt	5	6
INT1.7	38	0x0000 0D4C	2	TIMER0 interrupt	5	7
INT1.8	39	0x0000 0D4E	2	WAKE interrupt	5	8 (Lowest)
<b>PIE Group 2 Vectors - Muxed into CPU INT2</b>						
INT2.1	40	0x0000 0D50	2	EPWM1 trip zone interrupt	6	1 (Highest)
INT2.2	41	0x0000 0D52	2	EPWM2 trip zone interrupt	6	2
INT2.3	42	0x0000 0D54	2	EPWM3 trip zone interrupt	6	3
INT2.4	43	0x0000 0D56	2	EPWM4 trip zone interrupt	6	4
INT2.5	44	0x0000 0D58	2	EPWM5 trip zone interrupt	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6 trip zone interrupt	6	6
INT2.7	46	0x0000 0D5C	2	EPWM7 trip zone interrupt	6	7
INT2.8	47	0x0000 0D5E	2	Reserved	6	8 (Lowest)
<b>PIE Group 3 Vectors - Muxed into CPU INT3</b>						
INT3.1	48	0x0000 0D60	2	EPWM1 interrupt	7	1 (Highest)
INT3.2	49	0x0000 0D62	2	EPWM2 interrupt	7	2
INT3.3	50	0x0000 0D64	2	EPWM3 interrupt	7	3
INT3.4	51	0x0000 0D66	2	EPWM4 interrupt	7	4
INT3.5	52	0x0000 0D68	2	EPWM5 interrupt	7	5
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	Reserved	7	8 (Lowest)
<b>PIE Group 4 Vectors - Muxed into CPU INT4</b>						
INT4.1	56	0x0000 0D70	2	ECAP1 interrupt	8	1 (Highest)
INT4.2	57	0x0000 0D72	2	ECAP2 interrupt	8	2
INT4.3	58	0x0000 0D74	2	Reserved	8	3
INT4.4	59	0x0000 0D76	2	Reserved	8	4
INT4.5	60	0x0000 0D78	2	Reserved	8	5
INT4.6	61	0x0000 0D7A	2	Reserved	8	6
INT4.7	62	0x0000 0D7C	2	Reserved	8	7
INT4.8	63	0x0000 0D7E	2	Reserved	8	8 (Lowest)

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

Name	Vector ID	Address	Size (x16)	Description	Core Priority	ePIE Group priority
<b>PIE Group 5 Vectors - Muxed into CPU INT5</b>						
INT5.1	64	0x0000 0D80	2	EQEP1 interrupt	9	1 (Highest)
INT5.2	65	0x0000 0D82	2	Reserved	9	2
INT5.3	66	0x0000 0D84	2	Reserved	9	3
INT5.4	67	0x0000 0D86	2	Reserved	9	4
INT5.5	68	0x0000 0D88	2	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 (Lowest)
<b>PIE Group 6 Vectors - Muxed into CPU INT6</b>						
INT6.1	72	0x0000 0D90	2	SPIA RX interrupt	10	1 (Highest)
INT6.2	73	0x0000 0D92	2	SPIA TX interrupt	10	2
INT6.3	74	0x0000 0D94	2	Reserved	10	3
INT6.4	75	0x0000 0D96	2	Reserved	10	4
INT6.5	76	0x0000 0D98	2	Reserved	10	5
INT6.6	77	0x0000 0D9A	2	Reserved	10	6
INT6.7	78	0x0000 0D9C	2	DCC0 interrupt	10	7
INT6.8	79	0x0000 0D9E	2	Reserved	10	8 (Lowest)
<b>PIE Group 7 Vectors - Muxed into CPU INT7</b>						
INT7.1	80	0x0000 0DA0	2	Reserved	11	1 (Highest)
INT7.2	81	0x0000 0DA2	2	Reserved	11	2
INT7.3	82	0x0000 0DA4	2	Reserved	11	3
INT7.4	83	0x0000 0DA6	2	Reserved	11	4
INT7.5	84	0x0000 0DA8	2	Reserved	11	5
INT7.6	85	0x0000 0DAA	2	Reserved	11	6
INT7.7	86	0x0000 0DAC	2	Reserved	11	7
INT7.8	87	0x0000 0DAE	2	Reserved	11	8 (Lowest)
<b>PIE Group 8 Vectors - Muxed into CPU INT8</b>						
INT8.1	88	0x0000 0DB0	2	I2CA interrupt	12	1 (Highest)
INT8.2	89	0x0000 0DB2	2	I2CA FIFO interrupt	12	2
INT8.3	90	0x0000 0DB4	2	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	Reserved	12	7
INT8.8	95	0x0000 0DBE	2	Reserved	12	8 (Lowest)

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

Name	Vector ID	Address	Size (x16)	Description	Core Priority	ePIE Group priority
<b>PIE Group 9 Vectors - Muxed into CPU INT9</b>						
INT9.1	96	0x0000 0DC0	2	SCIA RX interrupt	13	1 (Highest)
INT9.2	97	0x0000 0DC2	2	SCIA TX interrupt	13	2
INT9.3	98	0x0000 0DC4	2	SCIB RX interrupt	13	3
INT9.4	99	0x0000 0DC6	2	SCIB TX interrupt	13	4
INT9.5	100	0x0000 0DC8	2	DCANA interrupt 1	13	5
INT9.6	101	0x0000 0DCA	2	DCANA interrupt 2	13	6
INT9.7	102	0x0000 0DCC	2	Reserved	13	7
INT9.8	103	0x0000 0DCE	2	Reserved	13	8 (Lowest)
<b>PIE Group 10 Vectors - Muxed into CPU INT10</b>						
INT10.1	104	0x0000 0DD0	2	ADCA event interrupt	14	1 (Highest)
INT10.2	105	0x0000 0DD2	2	ADCA2 interrupt	14	2
INT10.3	106	0x0000 0DD4	2	ADCA3 interrupt	14	3
INT10.4	107	0x0000 0DD6	2	ADCA4 interrupt	14	4
INT10.5	108	0x0000 0DD8	2	ADCC event interrupt	14	5
INT10.6	109	0x0000 0DDA	2	ADCC2 interrupt	14	6
INT10.7	110	0x0000 0DDC	2	ADCC3 interrupt	14	7
INT10.8	111	0x0000 0DDE	2	ADCC4	14	8 (Lowest)
<b>PIE Group 11 Vectors - Muxed into CPU INT11</b>						
INT11.1	112	0x0000 0DE0	2	Reserved	15	1 (Highest)
INT11.2	113	0x0000 0DE2	2	Reserved	15	2
INT11.3	114	0x0000 0DE4	2	Reserved	15	3
INT11.4	115	0x0000 0DE6	2	Reserved	15	4
INT11.5	116	0x0000 0DE8	2	Reserved	15	5
INT11.6	117	0x0000 0DEA	2	Reserved	15	6
INT11.7	118	0x0000 0DEC	2	Reserved	15	7
INT11.8	119	0x0000 0DEE	2	Reserved	15	8 (Lowest)
<b>PIE Group 12 Vectors - Muxed into CPU INT12</b>						
INT12.1	120	0x0000 0DF0	2	XINT3 interrupt	16	1 (Highest)
INT12.2	121	0x0000 0DF2	2	XINT4 interrupt	16	2
INT12.3	122	0x0000 0DF4	2	XINT5 interrupt	16	3
INT12.4	123	0x0000 0DF6	2	Reserved	16	4
INT12.5	124	0x0000 0DF8	2	FLSS_INT interrupt	16	5
INT12.6	125	0x0000 0DFA	2	Reserved	16	6
INT12.7	126	0x0000 0DFC	2	Reserved	16	7
INT12.8	127	0x0000 0DFE	2	Reserved	16	8 (Lowest)

## 3.6 Exceptions and Non-Maskable Interrupts

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

### 3.6.1 Configuring and Using NMIs

An incoming NMI sets a status bit in the NMIFLG register and starts the NMI watchdog counter. This counter is clocked by the SYSCLK, and if the counter reaches the value in the NMIWDPRD register, the counter 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 can also be cleared to allow future NMIs to be taken.

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

### 3.6.2 Emulation Considerations

The NMI watchdog counter behaves as follows under debug conditions:

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

### 3.6.3 NMI Sources

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

#### 3.6.3.1 Missing Clock Detection

The missing clock detection logic monitors OSCCLK for failure. If the OSCCLK source stops, the PLL is bypassed, OSCCLK is connected to INTOSC1, and an NMI is fired to the CPU. For more information on missing clock detection, see [Section 3.7.13.1](#).

#### 3.6.3.2 RAM Uncorrectable Error

A single-bit parity error, double-bit ECC data error, or single-bit ECC address error in a RAM read triggers an NMI. This applies to CPU reads. Single-bit ECC data errors do not trigger an NMI, but can optionally trigger a normal peripheral interrupt. For more information on RAM error detection, see [Section 3.11.1.4](#).

#### 3.6.3.3 Flash Uncorrectable ECC Error

A double-bit ECC data error or single-bit ECC address error in a Flash read triggers an NMI. Single-bit ECC data errors do not trigger an NMI, but can optionally trigger a normal peripheral interrupt.

#### 3.6.3.4 Software-Forced Error

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

### 3.6.4 Illegal Instruction Trap (ITRAP)

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

#### Note

A RAM fetch access violation triggers an ITRAP in addition to the normal peripheral interrupt for RAM access violations. The CPU handles the ITRAP first.

### 3.6.5 ERRORSTS Pin

The ERRORSTS pin is an 'always output' pin and remains high until an error is detected inside the chip. On an error, the ERRORSTS pin goes low (default polarity) until the corresponding internal error status flag for that error source is cleared. [Figure 3-4](#) shows the functionality of the ERRORSTS pin.

The ERRORSTS pin is tri-stated until the chip power rails ramp up to the lower operational limit. As the ERRORSTS pin is an active-low pin (default polarity), users who care about the state of this pin during power-up can connect an external pull-down on this pin.

Following enhancement has been made on this device for ERRORSTS pin logic:

- Polarity of Error pin has been made configurable through the ERRORCTL register (default setting is active-low polarity).
- To enable testing of the Error pin, capability to force and clear the Error pin from software has been provided.
- Additional sources of Error have been added to ERRORSTS:
  - CPU1 Watchdog reset
  - Error on a PIE vector fetch

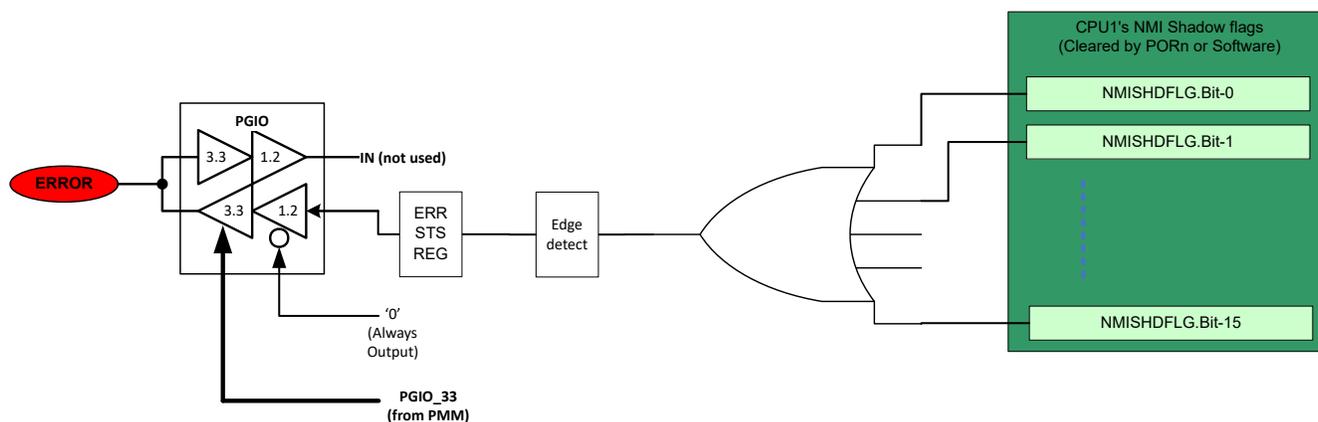


Figure 3-4. ERRORSTS Pin Diagram

## 3.7 Clocking

This section explains the clock sources and clock domains on this device, and how to configure them for application use. [Figure 3-5](#) and [Figure 3-6](#) provide an overview of the device's clocking system.

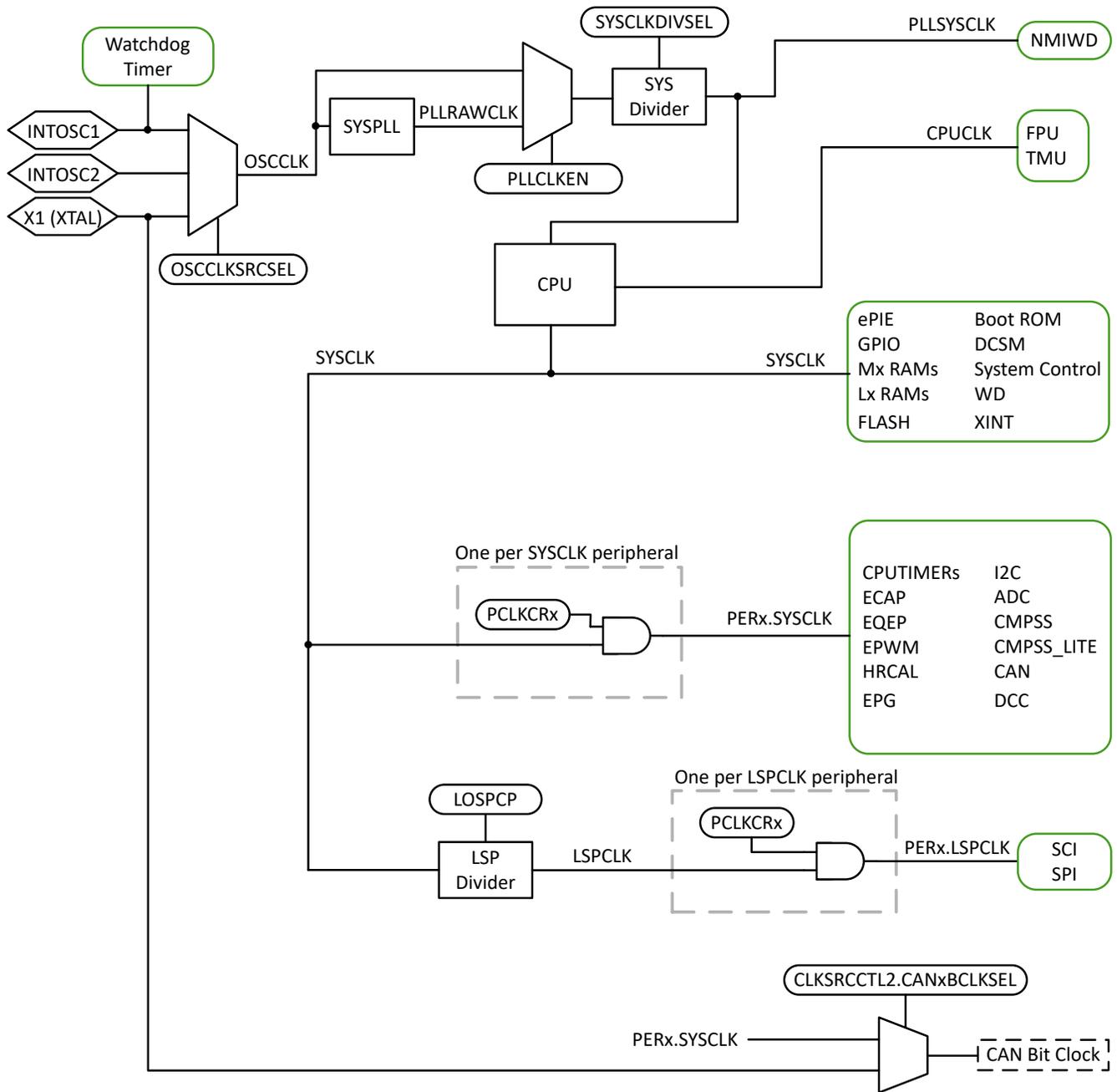
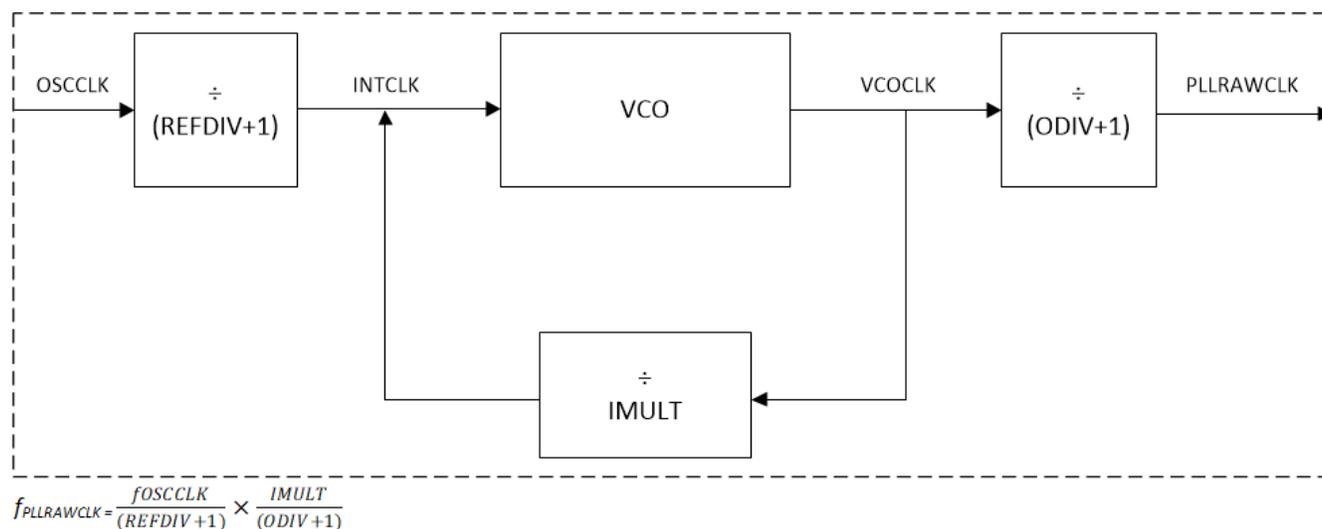


Figure 3-5. Clocking System

## SYSPLL



**Figure 3-6. System PLL**

### 3.7.1 Clock Sources

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

#### 3.7.1.1 Primary Internal Oscillator (INTOSC2)

At power-up, the device is clocked from an on-chip 10MHz oscillator (INTOSC2). INTOSC2 is the primary internal clock source and is the default system clock at reset. INTOSC2 is used to run the boot ROM and can be used as the system clock source for the application. Note that the INTOSC2 frequency tolerance is too loose to meet the timing requirements for CAN. Use of the CAN modules requires an external oscillator. When INTOSC2 is used as the system clock source, GPIO19 (X1) and GPIO18 (X2) are available as GPIO pins, unless ExtR functionality is used (GPIO19 is used for ExtR).

##### 3.7.1.1.1 External Resistor (ExtR) Mode

To improve the accuracy of INTOSC2, an external resistor (ExtR) mode can be used. This mode allows for a highly accurate external resistor to be added to the board and connected to the ExtR-capable pin. See the [TMS320F280013x Real-Time Microcontrollers Data Sheet](#) for which pin is ExtR-capable on each package.

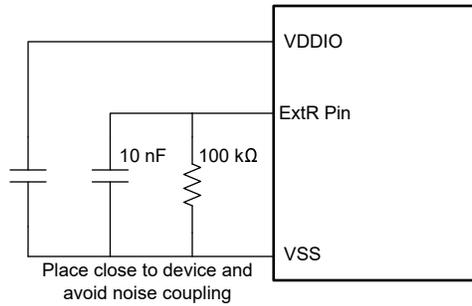
The INTOSC2 starts up in internal resistor mode, and therefore does not utilize the ExtR by default. When using the standard internal resistor, INTOSC2 is still fully functional, but with performance less than that of an accurate ExtR. This can be changed to use a more accurate external resistor (ExtR) by setting the CLK\_CFG\_REGS CLKSRCCTL1.INTOSC2CLKMODE bit to be 1. For more details on correctly enabling ExtR functionality, please see [Section 3.7.9](#).

##### 3.7.1.1.1.1 INTOSC2 with External Precision Resistor – ExtR

To achieve better accuracy, an external precision resistor can be used with INTOSC2. The external components required are:

- 100kΩ precision resistor between ExtR pin and VSS
- 10nF capacitor for noise filtering
- 20μF VDDIO capacitance minimum for low noise supply and load transients

[Figure 3-7](#) shows an example illustration of these required external components.



**Figure 3-7. ExtR Example Schematic**

In ExtR mode, the oscillator frequency error is directly proportional to the accuracy of the ExtR resistor.

The quality of the VDDIO supply directly affects the ExtR INTOSC performance. VDDIO capacitance values and circuit design must be decided with care to provide the cleanest supply possible to avoid jitter, noise, and other performance issues.

Placing a resistor on the ExtR pin prevents the pin from being used as a GPIO or X1.

Table 3-6 provides the ExtR specification values.

**Table 3-6. ExtR Specifications**

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
$f_{\text{INTOSC2-ExtR-ERR-PERC}}$	Ideal 0% error 100kΩ ExtR resistor	-0.7	0	+0.7	%
$f_{\text{INTOSC2-ExtR}}$	Ideal 0% error 100kΩ ExtR resistor	9.93	10	10.07	MHz
$f_{\text{ExtR-SETTLING}}$	Switch to ExtR Mode		1		ms
ExtR Resistance, $R_{\text{ExtR}}$			100		kΩ
ExtR Decoupling Capacitance, $C_{\text{ExtR}}$			10		nF
VDDIO Decoupling Capacitance, $C_{\text{VDDIO}}$		20			μF

Table 3-7 provides an example calculation for determining the total error of INTOSC2 given the parameters of a resistor.

**Table 3-7. Sample Total Error Calculation**

PARAMETER	VALUE	UNIT
INTOSC2 Ideal Frequency Variation	0.70	%
ExtR Resistor Tolerance	$R_{\text{TOLERANCE}}$	%
ExtR Resistor Temperature Coefficient	$R_{\text{TEMPCO}}$	ppm/°C
Operating Temperature	$T_{\text{OPERATING\_POINT}}$	°C
ExtR Data Sheet Ambient Temperature	$T_{\text{AMBIENT}}$	°C
Total Frequency Error	$\left[ \left( \frac{0.70}{100} \right) + \left( \frac{R_{\text{TOLERANCE}}}{100} \right) + \left( \frac{R_{\text{TEMPCO}}}{1E6} \right) * \text{abs}(T_{\text{OPERATING\_POINT}} - T_{\text{AMBIENT}}) \right] * 100$	%

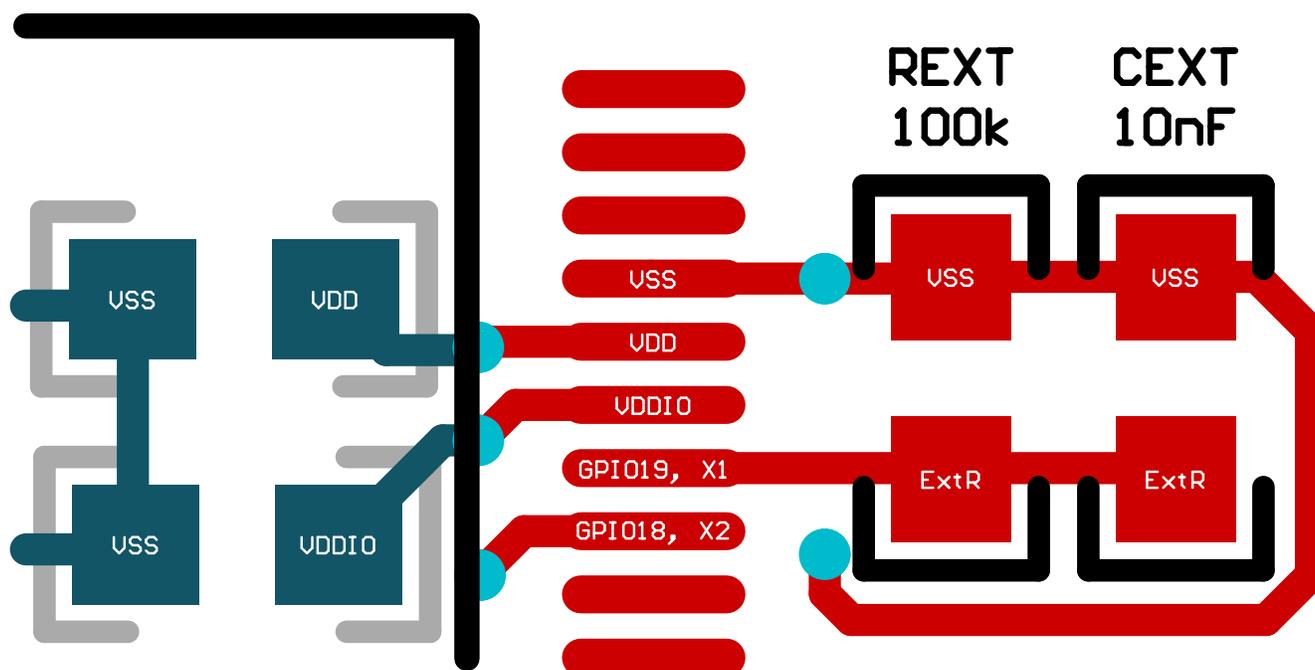
Table 3-8 provides example values using the above calculation.

**Table 3-8. Total Error Example Values**

PARAMETER	VALUE	UNIT
INTOSC2 Ideal Frequency Variation	0.70	%
ExtR Resistor Tolerance	0.10	%
ExtR Resistor Temperature Coefficient	25	ppm/°C
Operating Temperature	90	°C
ExtR Data Sheet Ambient Temperature	25	°C
Total Frequency Error Calculation	$((0.70/100) + (0.10/100) + ((25/1E6) * \text{abs}(90-25))) * 100$	%
Total Frequency Error Calculation	0.96	%

For best performance, use the following board layout guidelines:

- Route ExtR trace as short as possible
- Route ExtR to the nearest VSS pin
- Place ExtR ( $R_{\text{ExtR}}$ ) and  $C_{\text{ExtR}}$  on the same side as the C2000 device, with routing on the same layer only
- Any adjacent GPIO pin (GPIO18, X2 for example) can be routed using the opposite side and in a different layer so as to reduce adjacent GPIO coupling
- VSS connection must be tied both to VSS plane and directly to C2000 device VSS pin
- VSS guard trace is recommended around the ExtR trace as shown in [Figure 3-8](#)
- Fill VSS or VDDIO plane in layer below ExtR and  $C_{\text{ExtR}}$  to avoid routing signal traces in adjacent layer



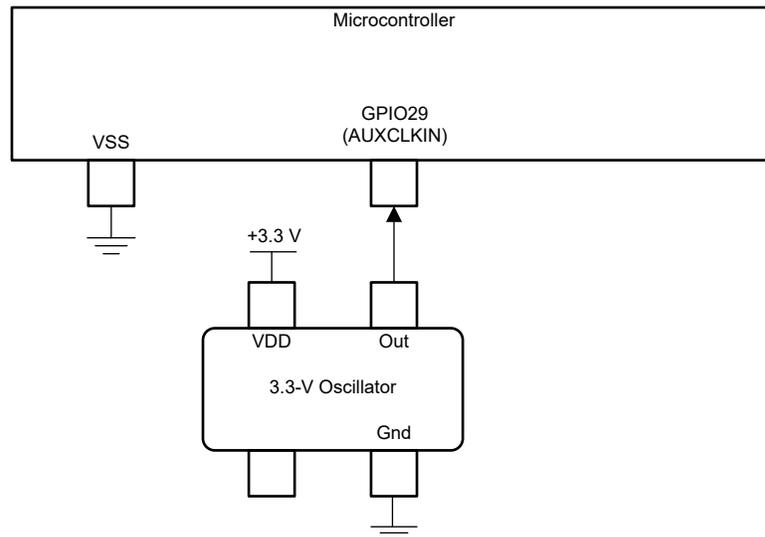
**Figure 3-8. ExtR PCB Layout Example**

### 3.7.1.2 Backup Internal Oscillator (INTOSC1)

The device also includes a redundant on-chip 10MHz 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 can also be manually selected as the system clock source for debug purposes.

### 3.7.1.3 Auxiliary Clock Input (AUXCLKIN)

An additional external clock source is supported on GPIO29 (AUXCLKIN). This must be a single-ended 3.3V external clock as shown in [Figure 3-9](#) and can be used as the clock source for DCAN and MCAN. Frequency limits and timing requirements are found in the [TMS320F28003x Real-Time Microcontrollers Data Sheet](#). The external clock can be connected directly to the GPIO29 pin.



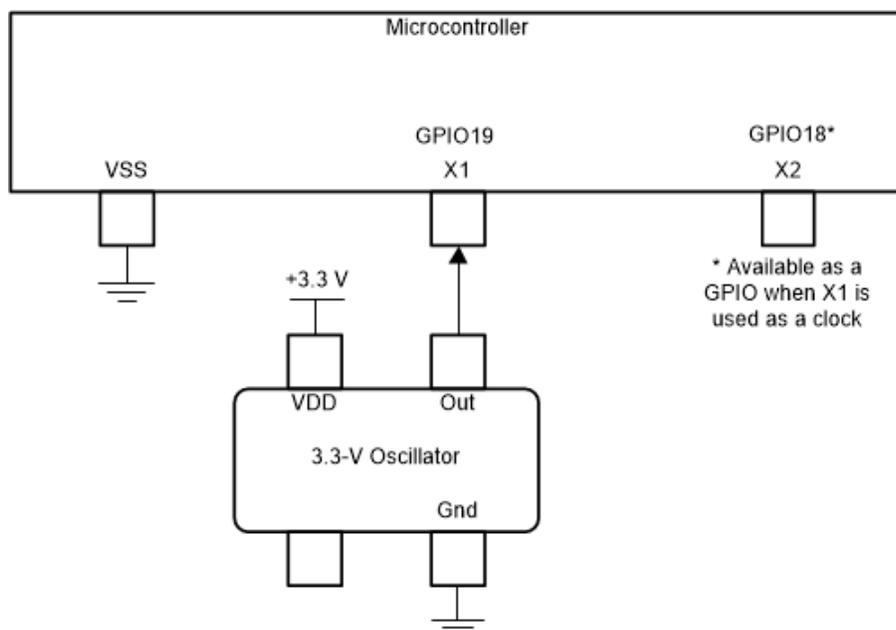
**Figure 3-9. AUXCLKIN**

### 3.7.1.4 External Oscillator (XTAL)

The device supports an external clock source (XTAL), which can be used as the main system and CAN bit clock source. Frequency limits and timing requirements can be found in the [TMS320F280013x Real-Time Microcontrollers Data Sheet](#). External clock sources use the X1/GPIO19 and X2/GPIO18 pins. After power-up, the X1 and X2 pin functionality can be enabled by following the procedure in [Section 3.7.6](#).

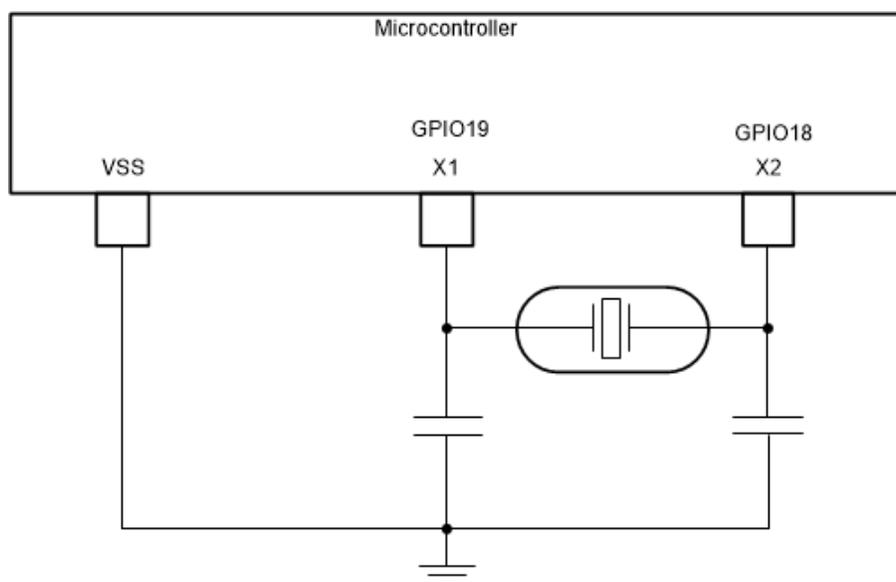
Three types of external clock sources are supported:

- A single-ended 3.3V external clock. The clock signal must be connected to X1, as shown in [Figure 3-10](#).



**Figure 3-10. Single-ended 3.3V External Clock**

- An external crystal. The crystal must be connected across X1 and X2 with the load capacitors connected to VSS, as shown in [Figure 3-11](#).



**Figure 3-11. External Crystal**

- An external resonator. The resonator must be connected across X1 and X2 with the ground connected to VSS, as shown in Figure 3-12.

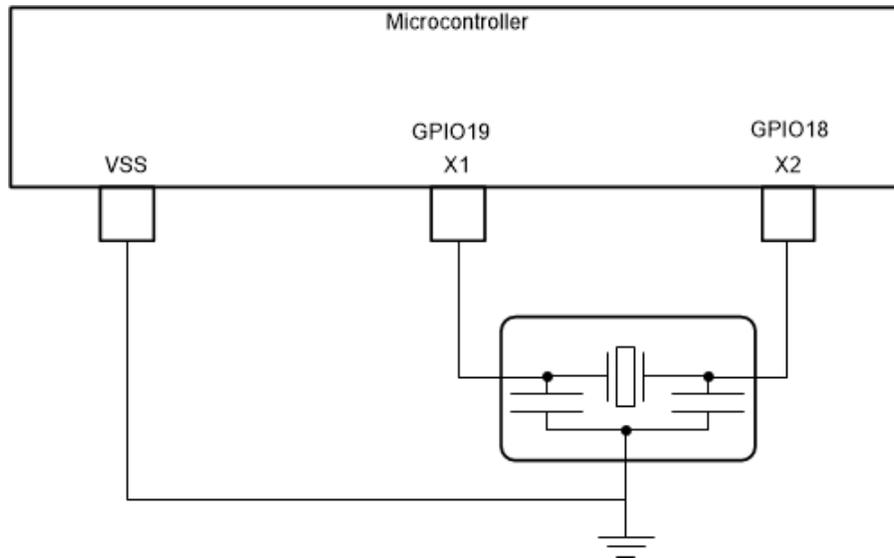


Figure 3-12. External Resonator

Table 3-9. ALT Modes

XTALCR Bit <sup>(1)</sup>		Operating Mode	GPIO19 Available on X1?	GPIO18 Available on X2?
OSCOFF	SE			
0	0	Crystal Mode: Quartz crystal connected to X1/X2	No	No
0	1	Single-Ended Mode: External clock on X1	No	Yes
1	0	Oscillator off	Yes	Yes
1	1	Single-Ended Mode: External clock on X1 <sup>(2)</sup>	No	Yes

(1) OSCOFF and SE determine the ALT mode of GPIO18 and GPIO19.

(2) There is an approximately 1KΩ pull-down on X1 in this mode, external single-ended clock must be able to drive this load.

### 3.7.2 Derived Clocks

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

#### 3.7.2.1 Oscillator Clock (OSCCLK)

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

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

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

### 3.7.3 Device Clock Domains

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

#### 3.7.3.1 System Clock (PLLSYSCLK)

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

#### 3.7.3.2 CPU Clock (CPUCLK)

The CPU has a clock (CPUCLK) that is used to clock the CPU and Flash wrapper. This clock is identical to PLLSYSCLK, but is gated when the CPU enters IDLE, STANDBY, or HALT mode.

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

The CPU provides a clock (SYSCLK) to the private (M0 and M1), local shared (LS0 and LS1), boot ROM and other peripherals. This clock is identical to PLLSYSCLK, but is gated when the CPU enters HALT or STANDBY mode.

Each peripheral clock has an independent clock gating that is controlled by the PCLKCRx registers.

---

#### Note

The application needs to wait for 5 SYSCLK cycles after enabling the clock to the peripherals when using PCLKCRx.

---

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

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

#### 3.7.3.5 CAN Bit Clock

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

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

#### 3.7.3.6 CPU Timer2 Clock (TIMER2CLK)

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

The main reason to use a non-SYSCLK source is for internal frequency measurement. In most applications, timer 2 runs off of SYSCLK.

### 3.7.4 XCLKOUT

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

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

### 3.7.5 Clock Connectivity

[Table 3-10](#) shows the clock connections sorted by the clock domain and [Table 3-11](#) shows the clock connections sorted by the module name.

**Table 3-10. Clock Connections Sorted by Clock Domain**

Clock Domain	Module Name
CPUCLK	FPU
	TMU
SYSCLK	ePIE
	Mx RAMs
	LSx RAMs
	Flash
	Boot ROM
	GPIO Input Sync and Qual
	WD
	XINT
	DCSM
	PLLSYSCLK
NMIWD	
PERx.SYSCLK	Timer0 - 2
	DCC0
	ePWM1 - 7
	eCAP1 - 2
	eQEP1
	ADCA, C
	CMPSS1
	CMPSS_LITE2 - 4
	I2CA - B
	CANA
	HRCAL
EPG	
PERx.LSPCLK	SCIA - C
	SPIA
WDCLK (INTOSC1)	Watchdog Timer

**Table 3-11. Clock Connections Sorted by Module Name**

Module Name	Clock Domain
ADCA, C	PERx.SYSCLK
Boot ROM	SYSCLK
CANA	PERx.SYSCLK
CMPSS1	PERx.SYSCLK
CMPSS_LITE2 - 4	PERx.SYSCLK
CPU	PLLSYSCLK
CPU Timers (0 - 2)	PERx.SYSCLK
DCC0	PERx.SYSCLK
DCSM	SYSCLK
eCAP1 - 2	PERx.SYSCLK
ePIE	SYSCLK
ePWM1 - 7	PERx.SYSCLK
eQEP1	PERx.SYSCLK
EPG	PERx.SYSCLK
Flash	SYSCLK
FPU	CPUCLK
GPIO Input Sync and Qual	SYSCLK
I2CA - B	PERx.SYSCLK
LSx RAMs	SYSCLK
Mx RAMs	SYSCLK
NMIWD	PLLSYSCLK
SCIA - C	PERx.LSPCLK
SPIA	PERx.LSPCLK
TMU	CPUCLK
Watchdog Timer	WDCLK (INTOSC1)

### 3.7.6 Clock Source and PLL Setup

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

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

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

### 3.7.7 Using an External Crystal or Resonator

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

1. Clear the XTALCR.OSCOFF bit.
2. Wait for the crystal to power up. 1ms is the typical wait time but this depends on the crystal that is being used.
3. Clear the X1 counter by writing a 1 to X1CNT.CLR and keep clearing until the X1 counter value in the X1CNT register is no longer saturated 2047 (0x7FF).
4. Wait for the X1 counter value in the X1CNT register to reach 2047 (0x7FF). Repeat steps 3-4 three additional times.
5. Select XTAL as the OSCCLK source by writing a 1 to CLKSRCCTL1.OSCCLKSRCSEL.
6. Check the MCLKSTS bit in the MCDPCR register. If the bit set, the oscillator has not finished powering up, and more time is required:
  - a. Clear the missing clock status by writing a 1 to MCDPCR.MCLKCLR.
  - b. Repeat steps 2-7. Do not reset the device. Doing so powers down the oscillator, which requires the procedure to be restarted from step 1.
  - c. If the oscillator has not finished powering up in 10 milliseconds, there is a real clock failure.
7. If MCDPCR.MCLKSTS is clear, the oscillator startup is a success. The system clock is now derived from XTAL.

### 3.7.8 Using an External Oscillator

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

1. Clear the XTALCR.OSCOFF bit.
2. Set the XTALCR.SE bit to enable single-ended mode.
3. Clear the X1 counter by writing a 1 to X1CNT.CLR and keep clearing until the X1 counter value in the X1CNT register is no longer saturated 2047 (0x7FF).
4. Wait for the X1 counter value in the X1CNT register to reach 2047 (0x7FF).
5. Repeat steps 3-4 three additional times.
6. Select XTAL as the OSCCLK source by writing a 1 to CLKSRCCTL1.OSCCLKSRCSEL.
7. Check the MCLKSTS bit in the MCDPCR register. If the bit set, either the external oscillator or the device has failed.
8. If MCLKSTS is clear, the switch to the external clock is a success. The system clock is now derived from XTAL.

### 3.7.9 Using an External Resistor (ExtR) With Internal Oscillator

INTOSC2 starts up in internal resistor mode. To enable the use of a precision external resistor, use the following steps:

1. Switch OSCCLKSRC source to INTOSC1 by setting CLKSRCCTL1.OSCCLKSRCSEL, followed by necessary delay for clock transition
2. Use driverlib function "ASysCtl\_setExtRCounterDelay(ASYCTL\_EXTR\_COUNTER\_DELAY\_2048);" to set required clock transition time.
3. Enable ExtR mode using driverlib function "SysCtl\_setIntOSC2\_Mode(SYSCTL\_INTOSC2\_MODE\_EXTR);", followed by a delay of 10000 cycles (can use "SysCtl\_delay(10000U);" function from driverlib to achieve a total 50009 cycles and provide additional buffer time).
4. Wait until "ASysCtl\_getExtROscStatus()" equals "ASYCTL\_EXTR\_ENABLE\_COMPLETE"
5. Switch OSCCLKSRC source back to INTOSC2 by setting CLKSRCCTL1.OSCCLKSRCSEL, followed by necessary delay for clock transition

After the steps above are completed, INTOSC2 uses the ExtR. Actions such as PLL locking must be initiated after all steps and included settling delays are complete.

### 3.7.10 Choosing PLL Settings

The equation shown in [Figure 3-6](#) can be used to configure the PLL.

IMULT is the integer value of the multiplier.

REFDIV is the reference divider for the OSCCLK.

ODIV is the output divider of the PLLRAWCLK.

PLLSYSCLKDIV is the system clock divider.

For the permissible values of the multipliers and dividers, see the documentation for the respective registers.

Many combinations of multiplier and divider can produce the same output frequency. However, the product of the reference clock frequency and the multiplier (known as the VCO frequency) must be in the range specified in the [TMS320F280013x Real-Time Microcontrollers Data Sheet](#) .

---

#### Note

The system clock frequency (PLLSYSCLK) can not exceed the limit specified in the [TMS320F280013x Real-Time Microcontrollers Data Sheet](#) . This limit does not allow for oscillator tolerance.

---

### 3.7.11 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:

Refer to your device SysCtl\_setClock() function inside C2000Ware installation for an example.

Recommended sequence to set up the system PLL:

1. Bypass the PLL by clearing SYSPLLCTL1[PLLCLKEN]. Allow at least 60 NOP instructions for this to take effect.
2. Power down the PLL by writing to SYSPLLCTL1.PPLEN = 0 and allow at least 60 NOP instructions for this to take effect.
3. Select the reference clock source (OSCCLK) by writing to CLKSRCCTL1.OSCCLKSRCSEL. Allow at least 300 NOP instructions for this to take effect.
4. Set the system clock divider to /1 to make sure the fastest PLL configuration by clearing SYSCLKDIVSEL[ PLLSYSCLKDIV].
5. Set the IMULT, REFDIV, and ODIV simultaneously by writing 32-bit value in SYSPLLMULT at once. This automatically enables the PLL. Be sure the settings for the multiplier and dividers do not violate the frequency specifications as defined in the [TMS320F280013x Real-Time Microcontrollers Data Sheet](#).
6. Wait for PLL to lock by polling for lock status bit to go high (SYSPLLSTS.LOCKS = 1).
7. Configure DCC with reference clock as OSCCLK and clock under measurement as PLLRAWCLK, and verify the frequency of the PLL. If the frequency is out of range, do not enable PLLRAWCLK as SYSCLK, stop here and troubleshoot. Refer to [Chapter 7](#) for more information on the configuration and usage.
8. Switch to the PLL as the system clock by setting SYSPLLCTL1[PLLCLKEN].

---

#### Note

1. SYSPLL must be bypassed and powered down manually before changing the OSCCLK source.
  2. At least 60 CPU clock cycles delay is needed after bypassing PLL, that is, SYSPLLCTL1.PLLCLKEN = 0.
  3. At least 60 CPU clock cycles delay is needed after PLL is powered down, that is, SYSPLLCTL1.PPLEN = 0.
  4. At least 300 CPU clock cycles delay is needed after OSSCLK source is changed.
  5. PLL SLIP bit is not supported. DCC can be used to check the validity of the PLL clock. This feature is included as part of SysCtl\_setClock() function inside C2000Ware.
- 

### 3.7.12 SYS PLL Bypass

If the application requires the PLL clock to be bypassed from the system, then the application needs to configure SYSPLLCTL1.PLLCLKEN = 0. It takes up to 60 CPU clock cycles before the bypass is effective. In the meantime if PLLSYSCLKDIV is reduced to a lower value (for example from /2 to /1, /4 to /2, and so on), the device can be clocked above the maximum rated frequency and can lead to unpredictable device behavior. Hence, a delay of 60 CPU clock cycles is required after bypassing the PLL from the enable state, that is, going from PLLCLKEN = 1 to PLLCLKEN = 0.

### 3.7.13 Clock (OSCCLK) Failure Detection

To achieve safety diagnostic, Missing Clock Detection (MCD) can be used.

Table 3-12 lists the details.

**Table 3-12. Clock Source (OSCCLK) Failure Detection**

Clock Failure Detection Circuitry	Clocks Detected	Time for Detection (in Cycles)	Limitations
Missing Clock Detection (MCD)	INTOSC2, XTAL/X1	8192 INTOSC1 cycles	Cannot detect INTOSC1 clock failure.

#### 3.7.13.1 Missing Clock Detection

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

- The primary clock (OSCCLK) clock keeps ticking a 7-bit counter (named as MCDPCNT). This counter is asynchronously reset with XRSn.
- The secondary clock (INTOSC1) clock keeps ticking a 13-bit counter (named as MCDSCNT). This counter is asynchronously reset with XRSn.
- 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 never overflows.
- If OSCCLK stops for some reason or is slower than INTOSC1 by at least a factor of 64, the MCDSCNT overflows and a missing clock condition is detected on OSCCLK.
- The above check is continuously active, unless the MCD is disabled using MCDPCR register (by making the MCLKOFF bit 1)
- 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.
  - PLL is forcefully bypassed and OSCCLK source is switched to INTOSC1 (New, System Clock Frequency = INTOSC1 Freq 10MHz)/SYSDIV). In the meantime when the clock switches to INTOSC1, the System runs on PLL limp Clock.
  - SYSPLLMULT.IMULT 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
- If the MCLKCLR bit is written (this is a W=1 bit), MCDSTS bit is cleared and OSCCLK source is decided by the OSCCLKSRCSEL bits. Writing to MCLKCLR also clears 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, switch the clock source to INTOSC1 (using OSCCLKSRCSEL register), do a MCLKCLR and re-lock the PLL.
- The MCD is enabled at power up.

Figure 3-13 shows the missing clock logic functional flow.

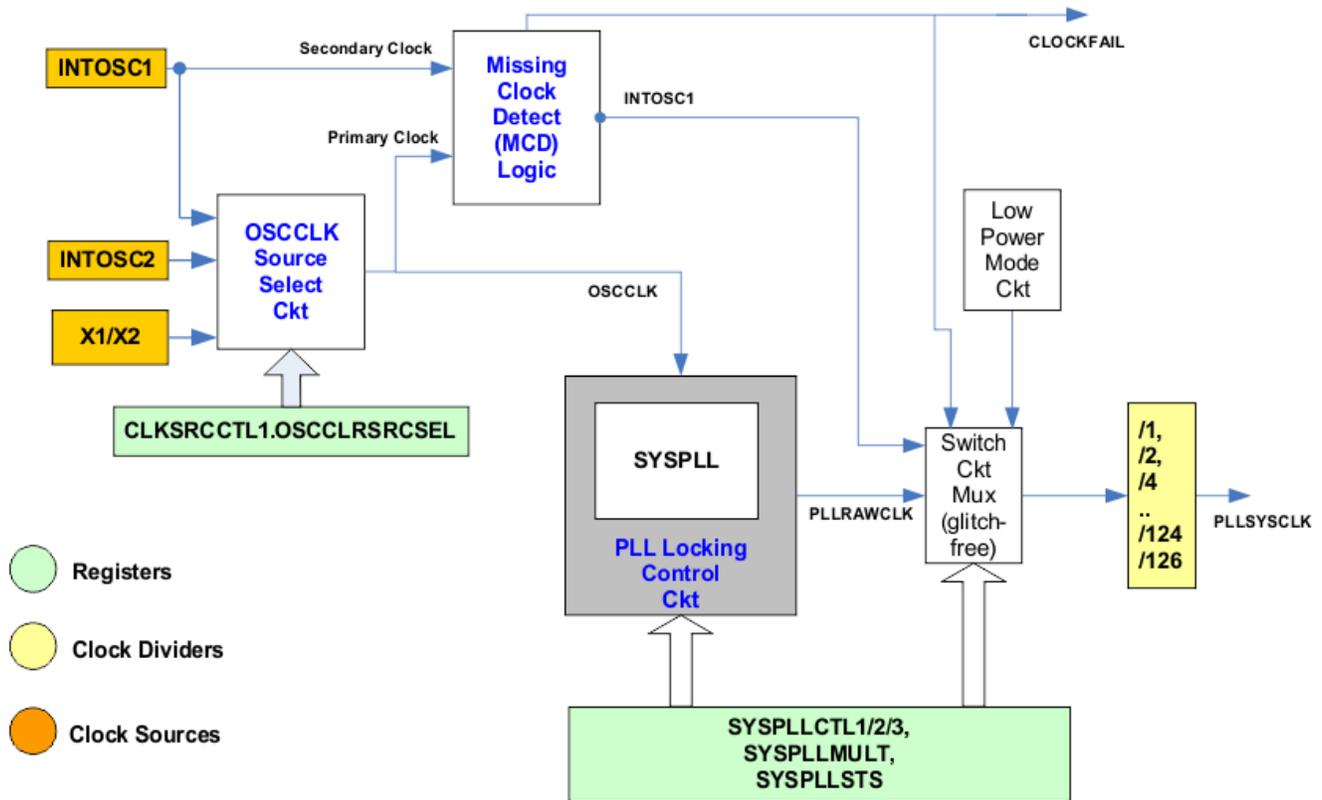


Figure 3-13. Missing Clock Detection Logic

**Note**

On a complete clock failure when OSCCLK is dead, it may take a maximum time of 8192 INTOSC1 cycles (that is, 0.8192ms) before the CLOCKFAIL signal goes high, after which:

- NMI is generated
- OSCCLK is switched to INTOSC1
- PWM Trip happens

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

This section describes the three 32-bit CPU timers (TIMER0/1/2) shown in Figure 3-14. Timer0 and Timer1 can be used in user applications. Timer2 is reserved for real-time operating system uses (for example, TI-RTOS). If the application is not using an operating system that utilizes this timer, then Timer2 can be used in the application. timer interrupt signals (TINT0, TINT1, TINT2) are connected as shown in Figure 3-15.

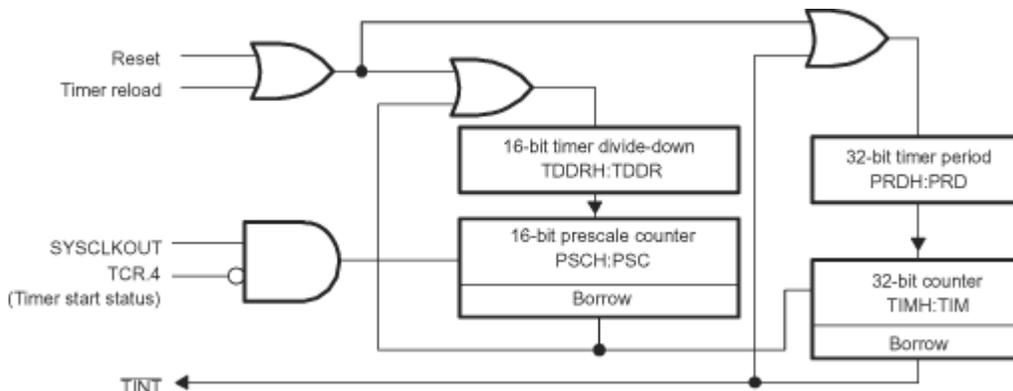
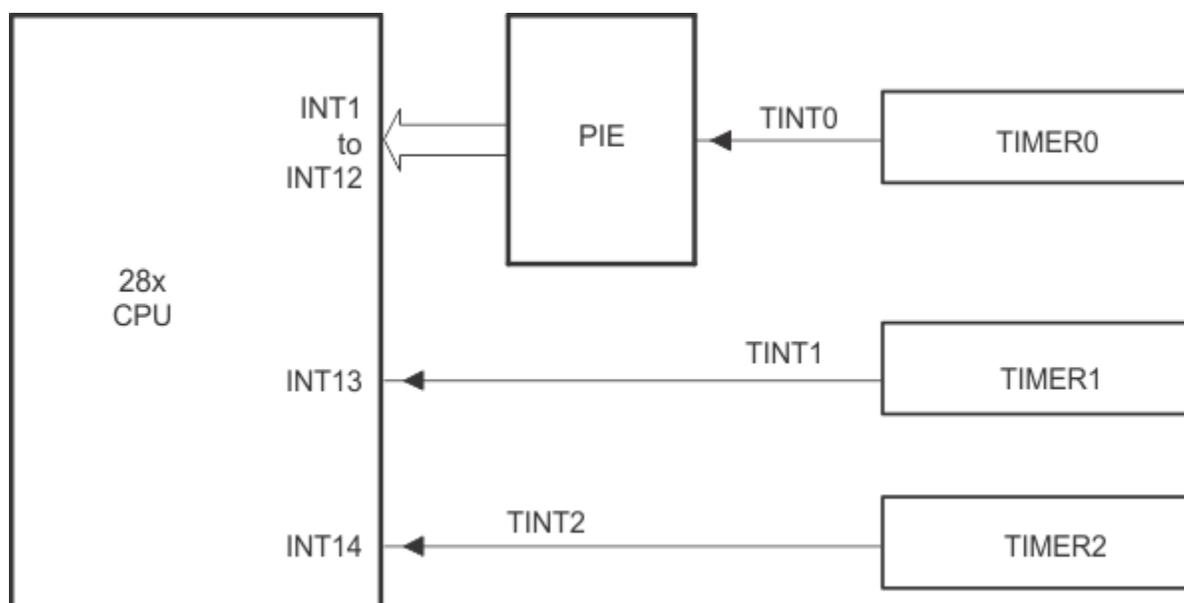


Figure 3-14. CPU Timers



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

Figure 3-15. CPU Timer Interrupt Signals and Output Signal

The general operation of a CPU timer is as follows:

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

The registers listed in Section 3.15 are used to configure the timers.

### 3.9 Watchdog Timer

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

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

Figure 3-16 shows the various functional blocks within the watchdog module.

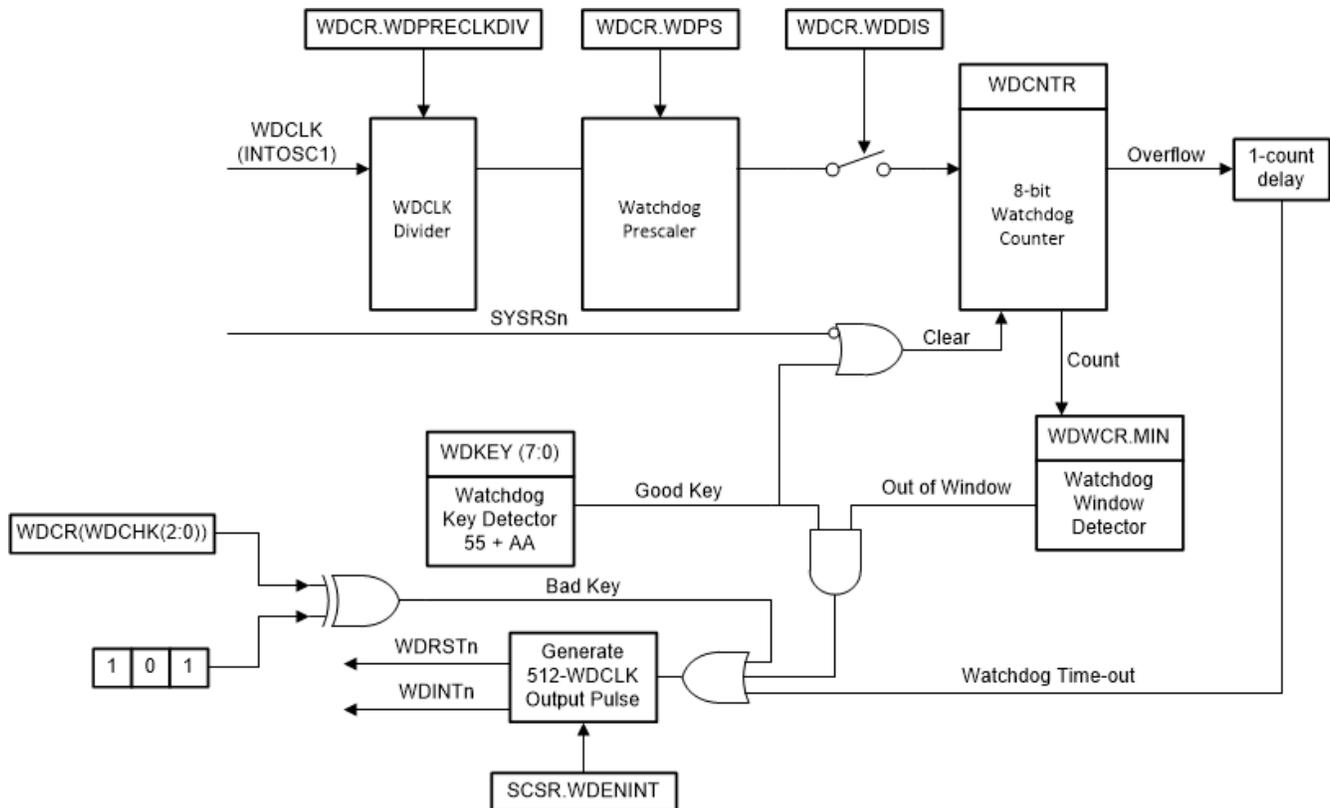


Figure 3-16. Watchdog Timer Module

### 3.9.1 Servicing the Watchdog Timer

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

The first action that enables the WDCNTR to be reset is shown in Step 3 in [Table 3-13](#). 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 resets 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 can clear WDRSn to allow subsequent watchdog resets to be detected. Watchdog resets are not prevented when the flag is set.

**Table 3-13. Example Watchdog Key Sequences**

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

### 3.9.2 Minimum Window Check

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

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

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

### 3.9.3 Watchdog Reset or Watchdog Interrupt Mode

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

- **Reset mode:** If the watchdog is configured to reset the device, then the  $\overline{WDRST}$  signal pulls the device reset ( $\overline{XRS}$ ) pin low for 512 INTOSC1 cycles when the watchdog counter reaches the maximum value.
- **Interrupt mode:** When the watchdog counter expires, an interrupt is asserted by driving the  $\overline{WDINT}$  signal low for 512 INTOSC1 cycles. The falling edge of  $\overline{WDINT}$  triggers a WAKEINT interrupt in the PIE if enabled. Because the PIE is edge-triggered, re-enabling the WAKEINT while  $\overline{WDINT}$  is active does not produce a duplicate interrupt.

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

### 3.9.4 Watchdog Operation in Low Power Modes

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

**Note:** If the watchdog interrupt is used to wake-up from an IDLE low power mode condition, software must make sure that the  $\overline{WDINT}$  signal goes back high before attempting to reenter the IDLE mode. The  $\overline{WDINT}$  signal is held low for 512 INTOSC1 cycles when the watchdog interrupt is generated. The current state of  $\overline{WDINT}$  can be determined by reading the watchdog interrupt status bit (WDINTS) bit in the SCSR register. WDINTS follows the state of  $\overline{WDINT}$  by two SYSCLKOUT cycles.

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

### 3.9.5 Emulation Considerations

The watchdog module behaves as follows under various debug conditions:

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

### 3.10 Low-Power Modes

This device has HALT, IDLE, and STANDBY as clock-gating low-power modes.

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

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

Before entering HALT mode check the value of the GPIODAT register of the pin selected for HALT wake-up (GPIOLPMSEL0/1) prior to entering the low-power mode to make sure that the wake event has not already been asserted.

#### 3.10.1 Clock-Gating Low-Power Modes

IDLE and HALT modes on this device are similar to those on other C28x devices. [Table 3-14](#) describes the effect on the system when any of the clock-gating low-power modes are entered.

**Table 3-14. Effect of Clock-Gating Low-Power Modes on the Device**

Modules/ Clock Domain	IDLE	STANDBY	HALT
SYSCLK	Active	Gated	Gated
CPUCLK	Gated	Gated	Gated
Clock to modules connected to PERx.SYSCLK	Active	Gated	Gated
WDCLK	Active	Active	Gated if CLKSRCCTL1.WDHALTI = 0
PLL	Powered	Powered	Software must power down PLL before entering HALT.
INTOSC1	Powered	Powered	Powered down if CLKSRCCTL1.WDHALTI = 0
INTOSC2	Powered	Powered	Powered down if CLKSRCCTL1.WDHALTI = 0
Flash <sup>(1)</sup>	Powered	Powered	Powered
XTAL <sup>(2)</sup>	Powered	Powered	Powered

- (1) The Flash module is not powered down by hardware in any LPM. It may be powered down using software if required by the application. For more information, see the *Flash Module* chapter.
- (2) The XTAL is not powered down by hardware in any LPM. It may be powered down by software setting the XTALCR.OSCOFF bit to 1. This can be done at any time during the application if the XTAL is not required.

#### 3.10.2 IDLE

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

Any enabled interrupt wakes up the CPU from IDLE mode.

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

The CPU resumes normal operations upon any enabled interrupt event.

### 3.10.3 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. STANDBY is best designed for an application where the wake-up signal comes from an external system (or CPU subsystem) rather than a peripheral input.

An NMI (or optionally, a watchdog interrupt or a configured GPIO) can wake the CPU from STANDBY mode. Each GPIO can be configured to wake the CPU when the GPIOs are driven active-low. Upon wakeup, the CPU receives the WAKEINT interrupt if configured.

#### 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; the signal 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 CPU is now out of STANDBY mode and can resume normal execution.

### 3.10.4 HALT

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

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

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

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

#### To enter HALT mode:

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

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

---

#### Note

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

---

#### To wake up from HALT mode:

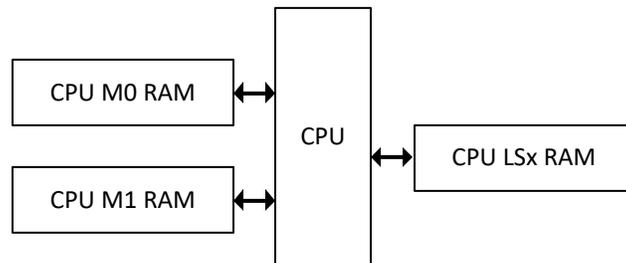
1. Drive the selected GPIO low for a minimum 5μs. This activates the WAKEINT PIE interrupt.
2. Drive the wake-up GPIO high again to initiate the powering up of the SYSPLL
3. Wait 16μs plus 1024 OSCLK cycles to allow the PLLs to lock and the WAKEINT ISR to be latched.
4. Execute the WAKEINT ISR.

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

### 3.11 Memory Controller Module

On this device, the M0, M1, and LSx RAMs are dedicated to the CPU.

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs are enabled with the parity feature (both data and address). Some of the dedicated memories are secure memory as well. Refer to [Chapter 5](#) for more details. Each RAM has a controller that takes care of the access protection/security related checks and ECC/Parity features for that RAM. [Figure 3-17](#) shows the configuration of these RAMs.



**Figure 3-17. Memory Architecture**

#### 3.11.1 Functional Description

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

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

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

All dedicated RAMs have the ECC feature.

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

Local shared RAMs (LSx RAMs) are secure memories and have Parity. These memories are dedicated to the CPU.

##### 3.11.1.3 Access Protection

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

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

---

#### Note

For debug accesses, all the protections are disabled.

---

### 3.11.1.3.1 CPU Fetch Protection

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

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

### 3.11.1.3.2 CPU Write Protection

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

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

### 3.11.1.3.3 CPU Read Protection

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

### 3.11.1.4 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 covers the data bits stored in memory as well as address.

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

#### 3.11.1.4.1 Error Detection and Correction

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

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

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

### 3.11.1.4.2 Error Handling

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

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

[Table 3-15](#) summarizes different error situations that can arise. These need to be handled appropriately in the software, using the status and interrupt indications provided.

**Table 3-15. Error Handling in Different Scenarios**

Access Type	Error Found In	Error Type	Status Indication	Error Notification
Reads	Data read from memory	Uncorrectable Error (Single-bit error for Parity RAMs or Double bit Error for ECC RAMs)	Yes - CPU Read Error Address Register Data returned to CPU is incorrect	NMI for CPU access
Reads	Data read from memory	Single-bit error for ECC RAMs	Yes - CPU Read Error Address Register Increment single error counter	Interrupt when error counter reaches the user programmable threshold for single errors
Reads	Data read from PIE memory	Parity error	Yes - PIE Parity Error sets bit in MEM_CFG_REGS	Bit set in MEM_CFG_REGS
Reads	Address	Address error	Yes - CPU Read Address Error Register Data returned to CPU is incorrect	NMI to CPU for CPU access

#### Note

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

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

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

Since error detection and correction logic is part of safety critical logic, safety applications need to make sure 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 can 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) must be done as 32-bit access only.

Table 3-16 and Table 3-17 show the bit mapping for the ECC/Parity bits when the bits are read in RAMTEST mode using the respective addresses.

**Table 3-16. Mapping of ECC Bits in Read Data from ECC/Parity Address Map**

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

**Table 3-17. Mapping of Parity Bits in Read Data from ECC/Parity Address Map**

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

### 3.11.1.6 RAM Initialization

To make sure 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, software can poll for the INITDONE bit for that RAM block in the INITDONE register to be set. Unless this bit gets set, no access must be made to that RAM memory block.

#### Note

None of the masters can access the memory while initialization is taking place. If memory is accessed before RAMINITDONE is set, the memory read/write as well as initialization does not happen correctly.

### 3.12 JTAG

Gel files perform certain initialization tasks. This helps the users in a debug environment. However, when executed standalone (without the emulator connected) the application can not work as expected, since there is no gel file to perform those initializations. For example, gel file disables watchdog. If user code does not service the watchdog in the application (or fails to disable the watchdog), there is a difference in how the application behaves with the debugger and without the debugger.

Common tasks performed by the gel files (but not boot-ROM):

- On Reset:
  - Disable Flash ECC on some devices.
    - Disabling ECC only when using Flash API functions, see the Flash API User Guide for details. Otherwise, TI suggests to always program ECC and enable ECC-check.
  - Disable Watchdog
  - Enable CLA clock
  - Select real-time mode or C28x mode
- On Restart:
  - Select real-time mode or C28x mode
  - Clear IER and IFR
- On Target Connect:
  - Select real-time mode or C28x mode

For more information, see [C2000 MCU JTAG Connectivity Debug Application Note](#).

#### 3.12.1 JTAG Noise and TAP\_STATUS

The TAP\_STATUS register reflects the status of the JTAG TAP at any given time. Normally when no JTAG is connected to the device, the status can be IDLE. In some cases with excessive PCB noise, there can be unwanted TMS and TCK toggles that take JTAG out of the IDLE state. When persistent, this can ultimately lead to unwanted activation of the JTAG Boundary Scan or some other JTAG mode that can interfere with the intended application. To avoid this scenario, place strong enough pull resistors on the board to prevent noise from activating JTAG. As a debug tool, the TAP\_STATUS register can be polled by the application code to detect if this is a cause of device disturbance. The SOFTPRES40[JTAG\_nTRST] register can also be used to reset the JTAG TAP through software. Use this reset register with caution, as this prevents connecting a debugger unless the code qualifies writes to this register with some other GPIO state or other means to distinguish between noise and debugger accesses.

### 3.13 System Control Register Configuration Restrictions

Memory-mapped registers in the System Control operate on INTOSC1 clock domain; hence, any CPU writes to these registers requires a delay between subsequent writes otherwise a second write can be lost. The application needs to take this into consideration and add a delay in terms of the number of NOP instructions after every write to these registers that are mentioned in [Table 3-18](#). The formula to compute delay between subsequent writes:

$$\text{Delay (in SYSCLK cycles)} = 3 \times (F_{\text{SYSCLK}} \div F_{\text{INTOSC1}}) + 9$$

For Example - For SYSCLK = 100MHz

$$\text{Delay (in SYSCLK cycles)} = 3 \times (100\text{MHz} \div 10\text{MHz}) + 9 = 39 \text{ SYSCLK cycles}$$

**Table 3-18. System Control Registers Impacted**

Registers requiring delay after every write
CLBCLKCTL
PERCLKDIVSEL
SYSCLKDIVSEL
SYSPLLCTL1
SYSPLLMULT
WDCR
XCLKOUTDIVSEL
XTALCR
CLKSRCCTL1
CLKSRCCTL2
CLKSRCCTL3
CPU1TMR2CTL (TMR2CLKCTL)

## 3.14 Software

### 3.14.1 SYSCTL Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/sysctl

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 3.14.1.1 Missing clock detection (MCD)

FILE: sysctl\_ex1\_missing\_clock\_detection.c

This example demonstrates the missing clock detection functionality and the way to handle it. Once the MCD is simulated by disconnecting the OSCCLK to the MCD module an NMI would be generated. This NMI determines that an MCD was generated due to a clock failure which is handled in the ISR.

Before an MCD the clock frequency would be as per device initialization (120Mhz). Post MCD the frequency would move to 10Mhz or INTOSC1.

The example also shows how we can lock the PLL after missing clock, detection, by first explicitly switching the clock source to INTOSC1, resetting the missing clock detect circuit and then re-locking the PLL. Post a re-lock the clock frequency would be 100Mhz but using the INTOSC1 as clock source.

##### External Connections

- None.

##### Watch Variables

- *fail* - Indicates that a missing clock was either not detected or was not handled correctly.
- *mcd\_clkfail\_isr* - Indicates that the missing clock failure caused an NMI to be triggered and called an the ISR to handle it.
- *mcd\_detect* - Indicates that a missing clock was detected.
- *result* - Status of a successful handling of missing clock detection

#### 3.14.1.2 XCLKOUT (External Clock Output) Configuration

FILE: sysctl\_ex2\_xclkout\_config.c

This example demonstrates how to configure the XCLKOUT pin for observing internal clocks through an external pin, for debugging and testing purposes.

In this example, we are using INTOSC1 as the XCLKOUT clock source and configuring the divider as 8. Expected frequency of XCLKOUT = (INTOSC1 freq)/8 = 10/8 = 1.25MHz

View the XCLKOUT on GPIO16 using an oscilloscope.

### 3.14.2 TIMER Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/timer

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 3.14.2.1 CPU Timers

FILE: timer\_ex1\_cputimers.c

This example configures CPU Timer0, 1, and 2 and increments a counter each time the timer asserts an interrupt. In order to migrate the project within syscfg to any device, click the swtich button under the device view and select your corresponding device to migrate, saving the project will auto-migrate your project settings.

##### External Connections

- None

##### Watch Variables

- cpuTimer0IntCount
- cpuTimer1IntCount
- cpuTimer2IntCount

### 3.14.2.2 CPU Timers

FILE: timer\_ex1\_cputimers\_sysconfig.c

This example configures CPU Timer0, 1, and 2 and increments a counter each time the timer asserts an interrupt.

#### External Connections

- None

#### Watch Variables

- cpuTimer0IntCount
- cpuTimer1IntCount
- cpuTimer2IntCount

### 3.14.3 MEMCFG Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/memcfg

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 3.14.3.1 Correctable & Uncorrectable Memory Error Handling

FILE: memcfg\_ex1\_error\_handling.c

This example demonstrates error handling in case of various erroneous memory read/write operations. Error handling in case of CPU read/write violations, correctable & uncorrectable memory errors has been demonstrated. Correctable memory errors & violations can generate SYS\_INT interrupt to CPU while uncorrectable errors lead to NMI generation.

#### External Connections

- None

#### Watch Variables

- *testStatusGlobal* - Equivalent to *TEST\_PASS* if test finished correctly, else the value is set to *TEST\_FAIL*
- *errCountGlobal* - Error counter

### 3.14.4 INTERRUPT Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/interrupt

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 3.14.4.1 External Interrupts (ExternalInterrupt)

FILE: interrupt\_ex1\_external.c

This program sets up GPIO0 as XINT1 and GPIO1 as XINT2. Two other GPIO signals are used to trigger the interrupt (GPIO10 triggers XINT1 and GPIO11 triggers XINT2). The user is required to externally connect these signals for the program to work properly.

XINT1 input is synced to SYSCLKOUT.

XINT2 has a long qualification - 6 samples at 510\*SYSCLKOUT each.

GPIO16 will go high outside of the interrupts and low within the interrupts. This signal can be monitored on a scope.

Each interrupt is fired in sequence - XINT1 first and then XINT2

### External Connections

- Connect GPIO10 to GPIO0. GPIO0 will be assigned to XINT1
- Connect GPIO11 to GPIO1. GPIO1 will be assigned to XINT2

Monitor GPIO16 with an oscilloscope. GPIO16 will be high outside of the ISRs and low within each ISR.

### Watch Variables

- xint1Count for the number of times through XINT1 interrupt
- xint2Count for the number of times through XINT2 interrupt
- loopCount for the number of times through the idle loop

### 3.14.4.2 Multiple interrupt handling of I2C, SCI & SPI Digital Loopback

FILE: interrupt\_ex2\_with\_i2c\_sci\_spi\_loopback.c

This program is used to demonstrate how to handle multiple interrupts when using multiple communication peripherals like I2C, SCI & SPI Digital Loopback all in a single example. The data transfers would be done with FIFO Interrupts.

It uses the internal loopback test mode of these modules. Both the TX and RX FIFOs and their interrupts are used. Other than boot mode pin configuration, no other hardware configuration is required.

A stream of data is sent and then compared to the received stream. The sent data looks like this for I2C and SCI:

```
0000 0001
0001 0002
0002 0003
...
00FE 00FF
00FF 0000
etc..
```

The sent data looks like this for SPI:

```
0000 0001
0001 0002
0002 0003
...
FFFE FFFF
FFFF 0000
etc..
```

This pattern is repeated forever.

### External Connections

- None

### Watch Variables

- *sDataI2cA* - Data to send through I2C
- *rDataI2cA* - Received I2C data
- *rDataPoint* - Used to keep track of the last position in the receive I2C stream for error checking
- *sDataSpiA* - Data to send through SPI
- *rDataSpiA* - Received SPI data
- *rDataPointSpiA* - Used to keep track of the last position in the receive SPI stream for error checking
- *sDataSciA* - SCI Data being sent
- *rDataSciA* - SCI Data received
- *rDataPointA* - Keep track of where we are in the SCI data stream. This is used to check the incoming data

### 3.14.4.3 CPU Timer Interrupt Software Prioritization

FILE: interrupt\_ex3\_sw\_prioritization.c

This examples demonstrates the software prioritization of interrupts through CPU Timer Interrupts. Software prioritization of interrupts is achieved by enabling interrupt nesting.

In this device, hardware priorities for CPU Timer 0, 1 and 2 are set as timer 0 being highest priority and timer 2 being lowest priority. This example configures CPU Timer0, 1, and 2 priority in software with timer 2 priority being highest and timer 0 being lowest in software and prints a trace for the order of execution.

For most applications, the hardware prioritizing of the interrupts is sufficient. For applications that need custom prioritizing, this example illustrates how this can be done through software. User specific priorities can be configured in `sw_prioritized_isr_level.h` header file.

To enable interrupt nesting, following sequence needs to followed in ISRs. *Step 1:* Set the global priority: Modify the IER register to allow CPU interrupts with a higher user priority to be serviced. Note: at this time IER has already been saved on the stack. *Step 2:* Set the group priority: (optional) Modify the appropriate PIEIERx register to allow group interrupts with a higher user set priority to be serviced. Do NOT clear PIEIER register bits from another group other than that being serviced by this ISR. Doing so can cause erroneous interrupts to occur. *Step 3:* Enable interrupts: There are three steps to do this: a. Clear the PIEACK bits b. Wait at least one cycle c. Clear the INTM bit. *Step 4:* Run the main part of the ISR *Step 5:* Set INTM to disable interrupts. *Step 6:* Restore PIEIERx (optional depending on step 2) *Step 7:* Return from ISR

Refer to below link on more details on Interrupt nesting in C28x devices:

[C2000Ware\docs\c28x\\_interrupt\\_nesting\html\index.html](http://<C2000Ware>\docs\c28x_interrupt_nesting\html\index.html)

#### External Connections

- None

#### Watch Variables

- `tracISR` - shows the order in which ISRs are executed.

#### 3.14.4.4 EPWM Real-Time Interrupt

FILE: `interrupt_ex4_epwm_realtime_interrupt.c`

This example configures the ePWM1 Timer and increments a counter each time the ISR is executed. ePWM interrupt can be configured as time critical to demonstrate real-time mode functionality and real-time interrupt capability.

The example uses 2 LEDs - LED1 is toggled in the main loop and LED2 is toggled in the EPWM Timer Interrupt. `FREE_SOFT` bits and `DBGIER.INT3` bit must be set to enable ePWM1 interrupt to be time critical and operational in real time mode after halt command

How to run the example?

- Add the watch variables as mentioned below and enable Continuous Refresh.
- Enable real-time mode (Run->Advanced->Enable Silicon Real-time Mode)
- Initially, the `DBGIER` register is set to 0 and the EPWM emulation mode is set to `EPWM_EMULATION_STOP_AFTER_NEXT_TB` (`FREE_SOFT = 0`)
- When the application is running, you will find both LEDs toggling and the watch variables `EPwm1TimerIntCount`, `EPwm1Regs.TBCTR` getting updated.
- When the application is halted, both LEDs stop toggling and the watch variables remain constant. EPWM counter is stopped on debugger halt.
- To enable EPWM counter run during debugger halt, set emulation mode as `EPWM_EMULATION_FREE_RUN` (`FREE_SOFT = 2`). You will find `EPwm1Regs.TBCTR` is running, but `EPwm1TimerIntCount` remains constant. This means, the EPWM counter is running, but the ISRs are not getting serviced.
- To enable real-time interrupts, set `DBGIER.INT3 = 1` (EPWM1 interrupt is part of PIE Group 3). You will find that the `EPwm1TimerIntCount` is incrementing and the LED starts toggling. The EPWM ISR is getting serviced even during a debugger halt.

For more details, watch this video : [C2000 Real-Time Features](#)

### External Connections

- None

### Watch Variables

- EPwm1TimerIntCount - EPWM1 ISR counter
- EPwm1Regs.TBCTR.TBCTR - EPWM1 Time Base counter
- EPwm1Regs.TBCTL.FREE\_SOFT - Set this to 2 to enable free run
- DBGIER.INT3 - Set to 1 to enable real time interrupt

## 3.14.5 LPM Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/lpm

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

### 3.14.5.1 Low Power Modes: Device Idle Mode and Wakeup using GPIO

FILE: lpm\_ex1\_idlewake\_gpio.c

This example puts the device into IDLE mode and then wakes up the device from IDLE using XINT1 which triggers on a falling edge of GPIO0.

The GPIO0 pin must be pulled from high to low by an external agent for wakeup. GPIO0 is configured as an XINT1 pin to trigger an XINT1 interrupt upon detection of a falling edge.

Initially, pull GPIO0 high externally. To wake device from IDLE mode by triggering an XINT1 interrupt, pull GPIO0 low (falling edge). The wakeup process begins as soon as GPIO0 is held low for the time indicated in the device datasheet.

GPIO1 is pulled high before entering the IDLE mode and is pulled low when in the external interrupt ISR.

#### External Connections

- GPIO0 needs to be pulled low to wake up the device.
- On device wakeup, the GPIO1 will be low and LED1 will start blinking

### 3.14.5.2 Low Power Modes: Device Idle Mode and Wakeup using Watchdog

FILE: lpm\_ex2\_idlewake\_watchdog.c

This example puts the device into IDLE mode and then wakes up the device from IDLE using watchdog timer.

The device wakes up from the IDLE mode when the watchdog timer overflows, triggering an interrupt. A pre scalar is set for the watchdog timer to change the counter overflow time.

GPIO1 is pulled high before entering the IDLE mode and is pulled low when in the wakeup ISR.

#### External Connections

- On device wakeup, the GPIO1 will be low and LED1 will start blinking

### 3.14.5.3 Low Power Modes: Device Standby Mode and Wakeup using GPIO

FILE: lpm\_ex3\_standbywake\_gpio.c

This example puts the device into STANDBY mode. If the lowest possible current consumption in STANDBY mode is desired, the JTAG connector must be removed from the device board while the device is in STANDBY mode.

This example puts the device into STANDBY mode and then wakes up the device from STANDBY using an LPM wakeup pin.

The pin GPIO0 is configured as the LPM wakeup pin to trigger a WAKEINT interrupt upon detection of a low pulse. Initially, pull GPIO0 high externally. To wake device from STANDBY mode, pull GPIO0 low for at least (2+QUALSTDBY), OSCLKS, then pull it high again.

The example then wakes up the device from STANDBY using GPIO0. GPIO0 wakes the device from STANDBY mode when a low pulse (signal goes high->low->high) is detected on the pin. This pin must be pulsed by an external agent for wakeup.

GPIO1 is pulled high before entering the STANDBY mode and is pulled low when in the wakeup ISR.

#### *External Connections*

- GPIO0 needs to be pulled low to wake up the device.
- On device wakeup, the GPIO1 will be low and LED1 will start blinking

#### **3.14.5.4 Low Power Modes: Device Standby Mode and Wakeup using Watchdog**

FILE: lpm\_ex4\_standbywake\_watchdog.c

This example puts the device into STANDBY mode. If the lowest possible current consumption in STANDBY mode is desired, the JTAG connector must be removed from the device board while the device is in STANDBY mode.

This example puts the device into STANDBY mode then wakes up the device from STANDBY using watchdog timer.

The device wakes up from the STANDBY mode when the watchdog timer overflows triggering an interrupt. In the ISR, the GPIO1 is pulled low. the GPIO1 is toggled to indicate the device is out of STANDBY mode. A pre scalar is set for the watchdog timer to change the counter overflow time.

GPIO1 is pulled high before entering the STANDBY mode and is pulled low when in the wakeup ISR.

#### *External Connections*

- On device wakeup, the GPIO1 will be low and LED1 will start blinking

#### **3.14.5.5 Low Power Modes: Halt Mode and Wakeup using GPIO**

FILE: lpm\_ex5\_haltwake\_gpio.c

This example puts the device into HALT mode. If the lowest possible current consumption in HALT mode is desired, the JTAG connector must be removed from the device board while the device is in HALT mode.

For applications that require minimal power consumption during HALT mode, application software should power off the XTAL prior to entering HALT by setting the XTALCR.OSCOFF bit or by using the driverlib function SysCtl\_turnOffOsc(SYSCTL\_OSCSRC\_XTAL);. If the OSCCLK source is configured to be XTAL, the application should first switch the OSSCLK source to INTOSC1 or INTOSC2 prior to setting XTALCR.OSCOFF.

This example puts the device into HALT mode and then wakes up the device from HALT using an LPM wakeup pin.

The pin GPIO0 is configured as the LPM wakeup pin to trigger a WAKEINT interrupt upon detection of a low pulse. The GPIO0 pin must be pulled from high to low by an external agent for wakeup.

GPIO1 is pulled high before entering the STANDBY mode and is pulled low when in the wakeup ISR.

#### *External Connections*

- On device wakeup, the GPIO1 will be low and LED1 will start blinking

#### **3.14.5.6 Low Power Modes: Halt Mode and Wakeup**

FILE: lpm\_ex6\_haltwake\_gpio\_watchdog.c

This example puts the device into HALT mode. If the lowest possible current consumption in HALT mode is desired, the JTAG connector must be removed from the device board while the device is in HALT mode.

For applications that require minimal power consumption during HALT mode, application software should power off the XTAL prior to entering HALT by setting the XTALCR.OSCOFF bit or by using the driverlib function SysCtl\_turnOffOsc(SYSCTL\_OSCSRC\_XTAL);. If the OSCCLK source is configured to be XTAL, the application should first switch the OSSCLK source to INTOSC1 or INTOSC2 prior to setting XTALCR.OSCOFF.

This example puts the device into HALT mode and then wakes up the device from HALT using an LPM wakeup pin.

The pin GPIO0 is configured as the LPM wakeup pin to trigger a WAKEINT interrupt upon detection of a low pulse. The GPIO0 pin must be pulled from high to low by an external agent for wakeup.

In this example, the watchdog timer is clocked, and is configured to produce watchdog reset as a timeout mechanism.

GPIO1 is pulled high before entering the STANDBY mode and is pulled low when in the wakeup ISR.

#### *External Connections*

- On device wakeup, the GPIO1 will be low and LED1 will start blinking

### **3.14.6 WATCHDOG Examples**

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/watchdog

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](#).

#### **3.14.6.1 Watchdog**

FILE: watchdog\_ex1\_service.c

This example shows how to service the watchdog or generate a wakeup interrupt using the watchdog. By default the example will generate a Wake interrupt. To service the watchdog and not generate the interrupt, uncomment the SysCtl\_serviceWatchdog() line in the main for loop.

#### *External Connections*

- None.

#### *Watch Variables*

- wakeCount - The number of times entered into the watchdog ISR
- loopCount - The number of loops performed while not in ISR

## 3.15 System Control Registers

### 3.15.1 SYSCTRL Base Address Table

**Table 3-19. SYSCTRL Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
CpuTimer0Regs	CPUTIMER_REGS	CPUTIMER0_BASE	0x0000_0C00	-
CpuTimer1Regs	CPUTIMER_REGS	CPUTIMER1_BASE	0x0000_0C08	-
CpuTimer2Regs	CPUTIMER_REGS	CPUTIMER2_BASE	0x0000_0C10	-
PieCtrlRegs	PIE_CTRL_REGS	PIECTRL_BASE	0x0000_0CE0	-
PieVectTable	PIE_VECT_TABLE	PIEVECTTABLE_BASE	0x0000_0D00	-
WdRegs	WD_REGS	WD_BASE	0x0000_7000	YES
NmiIntruptRegs	NMI_INTRUPT_REGS	NMI_BASE	0x0000_7060	YES
XintRegs	XINT_REGS	XINT_BASE	0x0000_7070	YES
SyncSocRegs	SYNC_SOC_REGS	SYNCSOC_BASE	0x0000_7940	YES
DevCfgRegs	DEV_CFG_REGS	DEVCFG_BASE	0x0005_D000	YES
ClkCfgRegs	CLK_CFG_REGS	CLKCFG_BASE	0x0005_D200	YES
CpuSysRegs	CPU_SYS_REGS	CPUSYS_BASE	0x0005_D300	YES
SysStatusRegs	SYS_STATUS_REGS	SYSSTAT_BASE	0x0005_D400	YES
MemCfgRegs	MEM_CFG_REGS	MEMCFG_BASE	0x0005_F400	YES
AccessProtectionRegs	ACCESS_PROTECTION_REGS	ACCESSPROTECTION_BASE	0x0005_F500	YES
MemoryErrorRegs	MEMORY_ERROR_REGS	MEMORYERROR_BASE	0x0005_F540	YES
TestErrorRegs	TEST_ERROR_REGS	TESTERROR_BASE	0x0005_F590	YES
UidRegs	UID_REGS	UID_BASE	0x0007_1140	-

### 3.15.2 ACCESS\_PROTECTION\_REGS Registers

Table 3-20 lists the memory-mapped registers for the ACCESS\_PROTECTION\_REGS registers. All register offset addresses not listed in Table 3-20 should be considered as reserved locations and the register contents should not be modified.

**Table 3-20. ACCESS\_PROTECTION\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
20h	MAVFLG	Master Access Violation Flag Register		<a href="#">Go</a>
22h	MAVSET	Master Access Violation Flag Set Register	EALLOW	<a href="#">Go</a>
24h	MAVCLR	Master Access Violation Flag Clear Register	EALLOW	<a href="#">Go</a>
26h	MAVINTEN	Master Access Violation Interrupt Enable Register	EALLOW	<a href="#">Go</a>
28h	MCPUFAVADDR	Master CPU Fetch Access Violation Address		<a href="#">Go</a>
2Ah	MCPUWRAVADDR	Master CPU Write Access Violation Address		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-21 shows the codes that are used for access types in this section.

**Table 3-21. ACCESS\_PROTECTION\_REGS Access Type Codes**

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

### 3.15.2.1 MAVFLG Register (Offset = 20h) [Reset = 0000000h]

MAVFLG is shown in [Figure 3-18](#) and described in [Table 3-22](#).

Return to the [Summary Table](#).

Master Access Violation Flag Register

**Figure 3-18. 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				RESERVED	RESERVED	CPUWRITE	CPUFETCH
R-0h				R-0h	R-0h	R-0h	R-0h

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

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	CPUWRITE	R	0h	Master CPU Write Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn
0	CPUFETCH	R	0h	Master CPU Fetch Access Violation Flag: 0: No violation. 1: Access violation occurred. Reset type: SYSRSn

### 3.15.2.2 MAVSET Register (Offset = 22h) [Reset = 0000000h]

MAVSET is shown in [Figure 3-19](#) and described in [Table 3-23](#).

Return to the [Summary Table](#).

Master Access Violation Flag Set Register

**Figure 3-19. 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				RESERVED	RESERVED	CPUWRITE	CPUFETCH
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

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

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn
0	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

### 3.15.2.3 MAVCLR Register (Offset = 24h) [Reset = 0000000h]

MAVCLR is shown in [Figure 3-20](#) and described in [Table 3-24](#).

Return to the [Summary Table](#).

Master Access Violation Flag Clear Register

**Figure 3-20. 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				RESERVED	RESERVED	CPUWRITE	CPUFETCH
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

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

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	CPUWRITE	R-0/W1S	0h	0: No action. 1: CPU Write Access Violation Flag in MAVFLG register will be cleared . Reset type: SYSRSn
0	CPUFETCH	R-0/W1S	0h	0: No action. 1: CPU Fetch Access Violation Flag in MAVFLG register will be cleared. Reset type: SYSRSn

### 3.15.2.4 MAVINTEN Register (Offset = 26h) [Reset = 0000000h]

MAVINTEN is shown in [Figure 3-21](#) and described in [Table 3-25](#).

Return to the [Summary Table](#).

Master Access Violation Interrupt Enable Register

**Figure 3-21. 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				RESERVED	RESERVED	CPUWRITE	CPUFETCH
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

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

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	CPUWRITE	R/W	0h	0: CPU Write Access Violation Interrupt is disabled. 1: CPU Write Access Violation Interrupt is enabled. Reset type: SYSRSn
0	CPUFETCH	R/W	0h	0: CPU Fetch Access Violation Interrupt is disabled. 1: CPU Fetch Access Violation Interrupt is enabled. Reset type: SYSRSn

### 3.15.2.5 MCPUFAVADDR Register (Offset = 28h) [Reset = 00000000h]

MCPUFAVADDR is shown in [Figure 3-22](#) and described in [Table 3-26](#).

Return to the [Summary Table](#).

Master CPU Fetch Access Violation Address

**Figure 3-22. MCPUFAVADDR Register**

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

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

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

### 3.15.2.6 MCPUWRAVADDR Register (Offset = 2Ah) [Reset = 0000000h]

MCPUWRAVADDR is shown in [Figure 3-23](#) and described in [Table 3-27](#).

Return to the [Summary Table](#).

Master CPU Write Access Violation Address

**Figure 3-23. MCPUWRAVADDR Register**

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

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

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

### 3.15.3 CLK\_CFG\_REGS Registers

Table 3-28 lists the memory-mapped registers for the CLK\_CFG\_REGS registers. All register offset addresses not listed in Table 3-28 should be considered as reserved locations and the register contents should not be modified.

**Table 3-28. CLK\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
2h	CLKCFGLOCK1	Lock bit for CLKCFG registers	EALLOW	<a href="#">Go</a>
8h	CLKSRCCTL1	Clock Source Control register-1	EALLOW	<a href="#">Go</a>
Ah	CLKSRCCTL2	Clock Source Control register-2	EALLOW	<a href="#">Go</a>
Ch	CLKSRCCTL3	Clock Source Control register-3	EALLOW	<a href="#">Go</a>
Eh	SYSPLLCTL1	SYSPLL Control register-1	EALLOW	<a href="#">Go</a>
14h	SYSPLLMULT	SYSPLL Multiplier register	EALLOW	<a href="#">Go</a>
16h	SYSPLLSTS	SYSPLL Status register		<a href="#">Go</a>
22h	SYSCLKDIVSEL	System Clock Divider Select register	EALLOW	<a href="#">Go</a>
28h	XCLKOUTDIVSEL	XCLKOUT Divider Select register	EALLOW	<a href="#">Go</a>
2Ch	LOSPCP	Low Speed Clock Source Prescaler	EALLOW	<a href="#">Go</a>
2Eh	MDCDR	Missing Clock Detect Control Register	EALLOW	<a href="#">Go</a>
30h	X1CNT	11-bit Counter on X1 Clock		<a href="#">Go</a>
32h	XTALCR	XTAL Control Register	EALLOW	<a href="#">Go</a>
3Ah	XTALCR2	XTAL Control Register for pad init	EALLOW	<a href="#">Go</a>
3Ch	CLKFAILCFG	Clock Fail cause Configuration	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-29 shows the codes that are used for access types in this section.

**Table 3-29. CLK\_CFG\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
WOnce	W Once	Write Set once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.

**Table 3-29. CLK\_CFG\_REGS Access Type Codes (continued)**

Access Type	Code	Description
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.15.3.1 CLKCFGLOCK1 Register (Offset = 2h) [Reset = 0000000h]

CLKCFGLOCK1 is shown in [Figure 3-24](#) and described in [Table 3-30](#).

Return to the [Summary Table](#).

Lock bit for CLKCFG registers

Notes:

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

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

**Figure 3-24. CLKCFGLOCK1 Register**

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

**Table 3-30. CLKCFGLOCK1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	RESERVED	R/WOnce	0h	Reserved
16	XTALCR	R/WOnce	0h	Common Lock bit for XTALCR & XTAL CR2 register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
15	LOSPCP	R/WOnce	0h	Lock bit for LOSPCP register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
14	RESERVED	R/WOnce	0h	Reserved
13	RESERVED	R/WOnce	0h	Reserved
12	RESERVED	R/WOnce	0h	Reserved
11	SYSCLKDIVSEL	R/WOnce	0h	Lock bit for SYSCLKDIVSEL register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
10	RESERVED	R/WOnce	0h	Reserved
9	RESERVED	R-0	0h	Reserved
8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/WOnce	0h	Reserved

**Table 3-30. CLKCFGLOCK1 Register Field Descriptions (continued)**

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

### 3.15.3.2 CLKSRCCTL1 Register (Offset = 8h) [Reset = 0000000h]

CLKSRCCTL1 is shown in [Figure 3-25](#) and described in [Table 3-31](#).

Return to the [Summary Table](#).

Clock Source Control register-1

This memory mapped register requires a delay of 45 SYCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-25. 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							RESERVED
R-0-0h							R/W-0h
7	6	5	4	3	2	1	0
INTOSC2CLKM ODE	RESERVED	WDHALTI	RESERVED	RESERVED	RESERVED	OSCCLKSRCSEL	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0-0h	R/W-0h	

**Table 3-31. CLKSRCCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-9	RESERVED	R-0	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	INTOSC2CLKMODE	R/W	0h	Selects between Internal Resistor mode and external Resistor mode for INTOSC2 0 - Internal resistor mode 1 - External resistor mode (ExtR) Reset type: XRSn
6	RESERVED	R/W	0h	Reserved
5	WDHALTI	R/W	0h	Watchdog HALT Mode Ignore Bit: This bit determines if WD is functional in the HALT mode or not. 0 = WD is not functional in the HALT mode. Clock to WD is gated when system enters HALT mode. 1 = WD is functional in the HALT mode. Clock to WD is not gated Reset type: XRSn
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R-0	0h	Reserved

**Table 3-31. 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)            01 = External Oscillator (XTAL)            10 = INTOSC1            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[13:0] 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:            [1] INTOSC1 is recommended to be used only after missing clock detection. If user wants to re-lock the PLL with INTOSC1 (the back-up clock source) after missing clock is detected, he can do a MCLKCLR and lock the PLL.</p> <p>Reset type: XRSn</p>

### 3.15.3.3 CLKSRCCTL2 Register (Offset = Ah) [Reset = 0000000h]

CLKSRCCTL2 is shown in [Figure 3-26](#) and described in [Table 3-32](#).

Return to the [Summary Table](#).

Clock Source Control register-2

This memory mapped register requires a delay of 45 SYSCCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-26. CLKSRCCTL2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
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		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		CANABCLKSEL		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-32. CLKSRCCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R/W	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R/W	0h	Reserved
3-2	CANABCLKSEL	R/W	0h	CANA Bit-Clock Source Select Bit: 00 = PERx.SYSCCLK (default on reset) 01 = External Oscillator (XTAL) 10 = AUXCLKIN (from GPIO) 11 = Reserved Missing clock detect circuit doesnt have any impact on these bits. Reset type: XRSn
1-0	RESERVED	R/W	0h	Reserved

### 3.15.3.4 CLKSRCCTL3 Register (Offset = Ch) [Reset = 0000000h]

CLKSRCCTL3 is shown in [Figure 3-27](#) and described in [Table 3-33](#).

Return to the [Summary Table](#).

Clock Source Control register-3

This memory mapped register requires a delay of 45 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-27. CLKSRCCTL3 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												XCLKOUTSEL			
R-0-0h												R/W-0h			

**Table 3-33. CLKSRCCTL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3-0	XCLKOUTSEL	R/W	0h	XCLKOUT Source Select Bit: This bit selects the source for XCLKOUT: 0x0 = PLLSYSCLK (default on reset) 0x1 = SYSPLLCLK 0x2 = SYSCLK 0x5 = INTOSC1 0x6 = INTOSC2 0x7 = XTAL OSC o/p clock 0xC = PLLRAWCLK Others = Reserved Reset type: SYSRSn

### 3.15.3.5 SYSPLLCTL1 Register (Offset = Eh) [Reset = 0000000h]

SYSPLLCTL1 is shown in [Figure 3-28](#) and described in [Table 3-34](#).

Return to the [Summary Table](#).

#### SYSPLL Control register-1

This memory mapped register requires a delay of 45 SYSCCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-28. 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		RESERVED	RESERVED	RESERVED	RESERVED	PLLCLKEN	PLLEN
R-0-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-34. SYSPLLCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	PLLCLKEN	R/W	0h	SYSPLL bypassed or included in the PLLSYSCLK path: This bit decides if the SYSPLL is bypassed when PLLSYSCLK is generated 1 = PLLSYSCLK is fed from the SYSPLL clock output. Users need to make sure that the PLL is locked before enabling this clock to the system. 0 = SYSPLL is bypassed. Clock to system is direct feed from OSCCLK Reset type: XRSn
0	PLLEN	R/W	0h	SYSPLL enabled or disabled: This bit decides if the SYSPLL is enabled or not 1 = SYSPLL is enabled 0 = SYSPLL is powered off. Clock to system is direct feed from OSCCLK Reset type: XRSn

### 3.15.3.6 SYSPLLMULT Register (Offset = 14h) [Reset = 0000000h]

SYSPLLMULT is shown in [Figure 3-29](#) and described in [Table 3-35](#).

Return to the [Summary Table](#).

SYSPLL Multiplier register

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

This memory mapped register requires a delay of 45 SYSCCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-29. SYSPLLMULT Register**

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

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

Bit	Field	Type	Reset	Description
31-29	RESERVED	R-0	0h	Reserved
28-24	REFDIV	R/W	0h	SYSPLL Reference Clock Divider PLL Reference Divider = REFDIV + 1 Reset type: XRSn
23-21	RESERVED	R-0	0h	Reserved
20-16	ODIV	R/W	0h	SYSPLL Output Clock Divider PLL Output Divider = ODIV + 1 ODIV should be set to 1 or greater to ensure the PLL output meets duty cycle requirements. Reset type: XRSn
15-14	RESERVED	R-0	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R-0	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-0	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 Note for APLL Multiplier values from 0-3 are invalid, internally those will be treated to 4. Reset type: XRSn

### 3.15.3.7 SYSPLLSTS Register (Offset = 16h) [Reset = 0000030h]

SYSPLLSTS is shown in [Figure 3-30](#) and described in [Table 3-36](#).

Return to the [Summary Table](#).

SYSPLL Status register

**Figure 3-30. 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		RESERVED	RESERVED	REF_LOSTS	RESERVED	SLIPS_NOTSU PPORTED	LOCKS
R-0-0h		R-1h	R-1h	W1C-0h	R-0h	R-0h	R-0h

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

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5	RESERVED	R	1h	Reserved
4	RESERVED	R	1h	Reserved
3	REF_LOSTS	W1C	0h	SYSPLL 'Reference Lost' Status Bit: This bit indicates whether the SYSPLL is out of lock range 0 = 'Reference Lost' event has not occurred. 1 = 'Reference Lost' event has occurred. Reset type: XRSn
2	RESERVED	R	0h	Reserved
1	SLIPS_NOTSUPPORTED	R	0h	RESERVED: This bit is reserved and the value read should be ignored. TI recommends using DCC to evaluate SYSPLL Slip status. Refer to InitSysPll() or SysCtl_setClock() functions inside the latest example software from C2000Ware for checking SYSPLL Slip status using DCC. Reset type: XRSn
0	LOCKS	R	0h	SYSPLL Lock Status Bit: This bit indicates whether the SYSPLL is locked or not 0 = SYSPLL is not yet locked 1 = SYSPLL is locked Reset type: XRSn

### 3.15.3.8 SYSCCLKDIVSEL Register (Offset = 22h) [Reset = 0000000h]

SYSCCLKDIVSEL is shown in [Figure 3-31](#) and described in [Table 3-37](#).

Return to the [Summary Table](#).

System Clock Divider Select register

This memory mapped register requires a delay of 45 SYSCCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-31. SYSCCLKDIVSEL 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							PLLSYSCCLKDI V_LSB
R-0-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED			PLLSYSCCLKDIV				
R-0-0h			R/W-0h				

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

Bit	Field	Type	Reset	Description
31-9	RESERVED	R-0	0h	Reserved
8	PLLSYSCCLKDIV_LSB	R/W	0h	This bit is LSB of the Divider that when set allows the ODD divisions such that the divider value is {PLLSYSCCLKDIV,PLLSYSCCLKDIV_LSB}. E.g. if PLLSYSCCLKDIV=0x1, and PLLSYSCCLKDIV_LSB=0 then divider of 2 is used else in case PLLSYSCCLKDIV_LSB=1 then divider value is 3. Reset type: XRSn
7-6	RESERVED	R-0	0h	Reserved
5-0	PLLSYSCCLKDIV	R/W	0h	PLLSYSCCLK Divide Select: This bit selects the divider setting for the PLLSYSCCLK. 000000 = /1 000001 = /2 000010 = /4 (default) 000011 = /6 000100 = /8 ..... 111111 = /126 Reset type: XRSn

### 3.15.3.9 XCLKOUTDIVSEL Register (Offset = 28h) [Reset = 0000003h]

XCLKOUTDIVSEL is shown in [Figure 3-32](#) and described in [Table 3-38](#).

Return to the [Summary Table](#).

XCLKOUT Divider Select register

This memory mapped register requires a delay of 45 SYCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-32. XCLKOUTDIVSEL Register**

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

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

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

### 3.15.3.10 LOSPCP Register (Offset = 2Ch) [Reset = 0000002h]

LOSPCP is shown in [Figure 3-33](#) and described in [Table 3-39](#).

Return to the [Summary Table](#).

Low Speed Clock Source Prescaler

**Figure 3-33. LOSPCP Register**

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

**Table 3-39. LOSPCP Register Field Descriptions**

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

### 3.15.3.11 MCDCCR Register (Offset = 2Eh) [Reset = 00006000h]

MCDCCR is shown in [Figure 3-34](#) and described in [Table 3-40](#).

Return to the [Summary Table](#).

Missing Clock Detect Control Register

**Figure 3-34. MCDCCR Register**

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

**Table 3-40. MCDCCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-15	RESERVED	R-0	0h	Reserved
14	RESERVED	R/W	1h	Reserved
13	RESERVED	R/W	1h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R	0h	Reserved
6	SYSREF_LOST_MCD_EN	R/W	0h	Control to add 'PLL reference clock lost' as cause for MCD 0 = 'PLL reference clock Lost' does not affect MCD. 1 = Upon 'PLL reference clock Lost' MCD is asserted. Reset type: XRSn
5	SYSREF_LOSTSCLR	R-0/W1S	0h	Clears the REF_LOST_STS from PLLSTS which is root for MCD trigger. 0 = No effect on present state of the REF_LOST_STS 1 = Clears the REF_LOST_STS bit to '0'. Bit clears itself after clear pulse to REF_LOST_STS. Read always gives '0'. Reset type: XRSn
4	SYSREF_LOSTS	R	0h	SYSPLL 'Reference Lost' Status Bit: This bit indicates whether the SYSPLL is out of lock range 0 = 'Reference Lost' event has not occurred. 1 = 'Reference Lost' event has occurred. Reset type: XRSn
3	OSCOFF	R/W	0h	Oscillator Clock Disconnect from MCD Bit: 0 = OSCCLK Connected to OSCCLK Counter in MCD module 1 = OSCCLK Disconnected to OSCCLK Counter in MCD module Reset type: XRSn

**Table 3-40. MCDCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	MCLKOFF	R/W	0h	Missing Clock Detect Off Bit: 0 = Missing Clock Detect Circuit Enabled 1 = Missing Clock Detect Circuit Disabled Reset type: XRSn
1	MCLKCLR	R-0/W1S	0h	Missing Clock Clear Bit: Write 1' to this bit to clear MCLKSTS bit and reset the missing clock detect circuit.' Reset type: XRSn
0	MCLKSTS	R	0h	Missing Clock Status Bit: 0 = OSCCLK Is OK 1 = OSCCLK Detected Missing, CLOCKFAILn Generated Reset type: XRSn

### 3.15.3.12 X1CNT Register (Offset = 30h) [Reset = 0000000h]

X1CNT is shown in [Figure 3-35](#) and described in [Table 3-41](#).

Return to the [Summary Table](#).

11-bit Counter on X1 Clock

**Figure 3-35. X1CNT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															CLR
R-0-0h															R-0/ W1C-0 h
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					X1CNT										
R-0-0h					R-0h										

**Table 3-41. X1CNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R-0	0h	Reserved
16	CLR	R-0/W1C	0h	X1 Counter clear: A write of '1' to this bit field clears the X1CNT and makes it count from 0x0 again (provided X1 clock is ticking). Writes of '0' are ignore to this bit field Reset type: XRSn
15-11	RESERVED	R-0	0h	Reserved
10-0	X1CNT	R	0h	X1 Counter: - This counter increments on every X1 CLOCKS positive-edge. - Once it reaches the values of 0x7ff, it freezes - Before switching from INTOSC2 to X1, application must check this counter and make sure that it has saturated. This will ensure that the Crystal connected to X1/X2 is oscillating. Reset type: XRSn

### 3.15.3.13 XTALCR Register (Offset = 32h) [Reset = 0000005h]

XTALCR is shown in [Figure 3-36](#) and described in [Table 3-42](#).

Return to the [Summary Table](#).

#### XTAL Control Register

This memory mapped register requires a delay of 45 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-36. XTALCR Register**

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

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

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2	RESERVED	R/W	1h	Reserved
1	SE	R/W	0h	Configures XTAL oscillator in single-ended or Crystal mode when XTAL oscillator is powered up (i.e. OSCOFF = 0) 0 XTAL oscillator in Crystal mode 1 XTAL oscillator in single-ended mode (through X1) Reset type: XRSn
0	OSCOFF	R/W	1h	This bit if '1', powers-down the XTAL oscillator macro and hence doesn't let X2 to be driven by the XTAL oscillator. If a crystal is connected to X1/X2, user needs to first clear this bit, wait for the oscillator to power up (using X1CNT) and then only switch the clock source to X1/X2 Reset type: XRSn

### 3.15.3.14 XTALCR2 Register (Offset = 3Ah) [Reset = 0000003h]

XTALCR2 is shown in [Figure 3-37](#) and described in [Table 3-43](#).

Return to the [Summary Table](#).

XTAL Control Register for pad init

**Figure 3-37. XTALCR2 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													FEN	XOF	XIF
R-0-0h													R/W-0h R/W-1h R/W-1h		

**Table 3-43. XTALCR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	0h	Reserved
15-3	RESERVED	R-0	0h	Reserved
2	FEN	R/W	0h	Configures XTAL oscillator pad initialisation. 0 : XOSC pads are not driven through GPIO connection. 1 : XOSC pads are driven through connected GPIO as per XIF & XOF values. This register has effect only when XOSC is OFF (no SE , no XTAL mode). If this register is set during XOSC off state (XOSCOFF=1 & SE=0) then upon change of these controls this bit gets reset and rearmed. Reset type: XRSn
1	XOF	R/W	1h	Polarity selection to initialise XO /X2 pad of the XOSC before start-up This value shall be deposited on the pad before XOSC started (XOSCOFF=1) If FEN=0 or XOSC is in XTAL or SE mode then this value will not be applied to the pad. Reset type: XRSn
0	XIF	R/W	1h	Polarity selection to initialise XI /X1 pad of the XOSC before start-up This value shall be deposited on the pad before XOSC started (XOSCOFF=1) If FEN=0 or XOSC is in XTAL or SE mode then this value will not be applied to the pad. Reset type: XRSn

### 3.15.3.15 CLKFAILCFG Register (Offset = 3Ch) [Reset = 0000000h]

CLKFAILCFG is shown in [Figure 3-38](#) and described in [Table 3-44](#).

Return to the [Summary Table](#).

Clock Fail cause Configuration

**Figure 3-38. CLKFAILCFG 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	DCC0_ERROR_EN
R-0-0h						R/W-0h	R/W-0h

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

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	DCC0_ERROR_EN	R/W	0h	This field enables DCC0 Error to cause the clock-fail NMI to get asserted. 0 : DCC0 Error does not affect Clock fail NMI 1: Occurrence of DCC0 Error triggers Clock fail NMI assertion and ERROR pin assertion. Reset type: XRSn

### 3.15.4 CPU\_SYS\_REGS Registers

Table 3-45 lists the memory-mapped registers for the CPU\_SYS\_REGS registers. All register offset addresses not listed in Table 3-45 should be considered as reserved locations and the register contents should not be modified.

**Table 3-45. CPU\_SYS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CPUSYSLOCK1	Lock bit for CPUSYS registers	EALLOW	<a href="#">Go</a>
2h	CPUSYSLOCK2	Lock bit for CPUSYS registers	EALLOW	<a href="#">Go</a>
Ah	PIEVERRADDR	PIE Vector Fetch Error Address register	EALLOW	<a href="#">Go</a>
22h	PCLKCR0	Peripheral Clock Gating Registers	EALLOW	<a href="#">Go</a>
26h	PCLKCR2	Peripheral Clock Gating Register - ETPWM	EALLOW	<a href="#">Go</a>
28h	PCLKCR3	Peripheral Clock Gating Register - ECAP	EALLOW	<a href="#">Go</a>
2Ah	PCLKCR4	Peripheral Clock Gating Register - EQEP	EALLOW	<a href="#">Go</a>
30h	PCLKCR7	Peripheral Clock Gating Register - SCI	EALLOW	<a href="#">Go</a>
32h	PCLKCR8	Peripheral Clock Gating Register - SPI	EALLOW	<a href="#">Go</a>
34h	PCLKCR9	Peripheral Clock Gating Register - I2C	EALLOW	<a href="#">Go</a>
36h	PCLKCR10	Peripheral Clock Gating Register - CAN	EALLOW	<a href="#">Go</a>
3Ch	PCLKCR13	Peripheral Clock Gating Register - ADC	EALLOW	<a href="#">Go</a>
3Eh	PCLKCR14	Peripheral Clock Gating Register - CMPSS	EALLOW	<a href="#">Go</a>
4Ch	PCLKCR21	Peripheral Clock Gating Register - DCC	EALLOW	<a href="#">Go</a>
58h	PCLKCR27	Peripheral Clock Gating Register - EPG	EALLOW	<a href="#">Go</a>
70h	SIMRESET	Simulated Reset Register		<a href="#">Go</a>
76h	LPMCR	LPM Control Register	EALLOW	<a href="#">Go</a>
78h	GPIOLPMSEL0	GPIO LPM Wakeup select registers	EALLOW	<a href="#">Go</a>
7Ah	GPIOLPMSEL1	GPIO LPM Wakeup select registers	EALLOW	<a href="#">Go</a>
7Ch	TMR2CLKCTL	Timer2 Clock Measurement functionality control register	EALLOW	<a href="#">Go</a>
7Eh	RESCCLR	Reset Cause Clear Register		<a href="#">Go</a>
80h	RESC	Reset Cause register		<a href="#">Go</a>
A0h	USER_REG1_SYRSn	Software Configurable registers reset by SYRSn		<a href="#">Go</a>
A2h	USER_REG2_SYRSn	Software Configurable registers reset by SYRSn		<a href="#">Go</a>
A4h	USER_REG1_XRSn	Software Configurable registers reset by XRSn		<a href="#">Go</a>
A6h	USER_REG2_XRSn	Software Configurable registers reset by XRSn		<a href="#">Go</a>
A8h	USER_REG1_PORESETn	Software Configurable registers reset by PORESETn		<a href="#">Go</a>
AAh	USER_REG2_PORESETn	Software Configurable registers reset by PORESETn		<a href="#">Go</a>
ACh	USER_REG3_PORESETn	Software Configurable registers reset by PORESETn		<a href="#">Go</a>
A Eh	USER_REG4_PORESETn	Software Configurable registers reset by PORESETn		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-46 shows the codes that are used for access types in this section.

**Table 3-46. CPU\_SYS\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read

**Table 3-46. CPU\_SYS\_REGS Access Type Codes (continued)**

Access Type	Code	Description
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
WOnce	W Once	Write Set once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.15.4.1 CPUSYSLOCK1 Register (Offset = 0h) [Reset = 0000000h]

CPUSYSLOCK1 is shown in [Figure 3-39](#) and described in [Table 3-47](#).

Return to the [Summary Table](#).

Lock bit for CPUSYS registers

Notes:

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

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

**Figure 3-39. CPUSYSLOCK1 Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	PCLKCR22	PCLKCR21	PCLKCR20	RESERVED	RESERVED	RESERVED
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
GPIOLPMSEL1	GPIOLPMSEL0	LPMCR	RESERVED	RESERVED	RESERVED	PCLKCR14	PCLKCR13
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
RESERVED	RESERVED	PCLKCR10	PCLKCR9	PCLKCR8	PCLKCR7	PCLKCR6	RESERVED
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
PCLKCR4	PCLKCR3	PCLKCR2	RESERVED	PCLKCR0	PIEVERRADDR	RESERVED	RESERVED
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-47. CPUSYSLOCK1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	PCLKCR22	R/WOnce	0h	Lock bit for PCLKCR22 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
28	PCLKCR21	R/WOnce	0h	Lock bit for PCLKCR21 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
27	PCLKCR20	R/WOnce	0h	Lock bit for PCLKCR20 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
26	RESERVED	R/WOnce	0h	Reserved
25	RESERVED	R/WOnce	0h	Reserved
24	RESERVED	R/WOnce	0h	Reserved
23	GPIOLPMSEL1	R/WOnce	0h	Lock bit for GPIOLPMSEL1 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
22	GPIOLPMSEL0	R/WOnce	0h	Lock bit for GPIOLPMSEL0 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn

**Table 3-47. CPUSYSLOCK1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
21	LPMCR	R/WOnce	0h	Lock bit for LPMCR Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
20	RESERVED	R/WOnce	0h	Reserved
19	RESERVED	R/WOnce	0h	Reserved
18	RESERVED	R/WOnce	0h	Reserved
17	PCLKCR14	R/WOnce	0h	Lock bit for PCLKCR14 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
16	PCLKCR13	R/WOnce	0h	Lock bit for PCLKCR13 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
15	RESERVED	R/WOnce	0h	Reserved
14	RESERVED	R/WOnce	0h	Reserved
13	PCLKCR10	R/WOnce	0h	Lock bit for PCLKCR10 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
12	PCLKCR9	R/WOnce	0h	Lock bit for PCLKCR9 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
11	PCLKCR8	R/WOnce	0h	Lock bit for PCLKCR8 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
10	PCLKCR7	R/WOnce	0h	Lock bit for PCLKCR7 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
9	PCLKCR6	R/WOnce	0h	Lock bit for PCLKCR6 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
8	RESERVED	R/WOnce	0h	Reserved
7	PCLKCR4	R/WOnce	0h	Lock bit for PCLKCR4 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
6	PCLKCR3	R/WOnce	0h	Lock bit for PCLKCR3 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
5	PCLKCR2	R/WOnce	0h	Lock bit for PCLKCR2 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
4	RESERVED	R/WOnce	0h	Reserved

**Table 3-47. CPUSYSLOCK1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	PCLKCR0	R/WOnce	0h	Lock bit for PCLKCR0 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
2	PIEVERRADDR	R/WOnce	0h	Lock bit for PIEVERRADDR Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
1	RESERVED	R/WOnce	0h	Reserved
0	RESERVED	R/WOnce	0h	Reserved

### 3.15.4.2 CPUSYSLOCK2 Register (Offset = 2h) [Reset = 0000000h]

CPUSYSLOCK2 is shown in [Figure 3-40](#) and described in [Table 3-48](#).

Return to the [Summary Table](#).

Lock bit for CPUSYS registers

Notes:

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

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

**Figure 3-40. CPUSYSLOCK2 Register**

31	30	29	28	27	26	25	24
USER_REG4_PORESETn	USER_REG3_PORESETn	USER_REG2_PORESETn	USER_REG1_PORESETn	USER_REG2_XRSn	USER_REG1_XRSn	USER_REG2_SYRSn	USER_REG1_SYRSn
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
RESERVED	RESERVED						
R/WOnce-0h	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	PCLKCR27	RESERVED	RESERVED	RESERVED
R-0-0h			R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-48. CPUSYSLOCK2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	USER_REG4_PORESETn	R/WOnce	0h	Lock bit for USER_REG4_PORESETn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
30	USER_REG3_PORESETn	R/WOnce	0h	Lock bit for USER_REG3_PORESETn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
29	USER_REG2_PORESETn	R/WOnce	0h	Lock bit for USER_REG2_PORESETn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
28	USER_REG1_PORESETn	R/WOnce	0h	Lock bit for USER_REG1_PORESETn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
27	USER_REG2_XRSn	R/WOnce	0h	Lock bit for USER_REG2_XRSn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
26	USER_REG1_XRSn	R/WOnce	0h	Lock bit for USER_REG1_XRSn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn

**Table 3-48. CPUSYSLOCK2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25	USER_REG2_SYSRSn	R/WOnce	0h	Lock bit for USER_REG2_SYSRSn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
24	USER_REG1_SYSRSn	R/WOnce	0h	Lock bit for USER_REG1_SYSRSn Register 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
23	RESERVED	R/WOnce	0h	Reserved
22-5	RESERVED	R-0	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	PCLKCR27	R/WOnce	0h	Lock bit for PCLKCR27 Register: 0: Respective register is not locked 1: Respective register is locked. Reset type: SYSRSn
2	RESERVED	R/WOnce	0h	Reserved
1	RESERVED	R/WOnce	0h	Reserved
0	RESERVED	R/WOnce	0h	Reserved

### 3.15.4.3 PIEVERRADDR Register (Offset = Ah) [Reset = 003FFFFFFh]

PIEVERRADDR is shown in [Figure 3-41](#) and described in [Table 3-49](#).

Return to the [Summary Table](#).

PIE Vector Fetch Error Address register

**Figure 3-41. PIEVERRADDR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										ADDR																					
R-0-0h										R/W-003FFFFFFh																					

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

Bit	Field	Type	Reset	Description
31-22	RESERVED	R-0	0h	Reserved
21-0	ADDR	R/W	003FFFFFFh	This register defines the address of the PIE Vector Fetch Error handler routine. Its the responsibility of user to initialize this register. If this register is not initialized, a default error handler at address 0x3ffbe will get executed. Refer to the Boot ROM section for more details on this register. Reset type: XRSn

### 3.15.4.4 PCLKCR0 Register (Offset = 22h) [Reset = 0000038h]

PCLKCR0 is shown in [Figure 3-42](#) and described in [Table 3-50](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Registers

**Figure 3-42. PCLKCR0 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	TBCLKSYNC	RESERVED	HRCAL
R-0-0h				R/W-0h	R/W-0h	R-0-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED				
R-0-0h	R/W-0h	R/W-0h	R-0-0h				
7	6	5	4	3	2	1	0
RESERVED		CPUTIMER2	CPUTIMER1	CPUTIMER0	RESERVED	RESERVED	RESERVED
R-0-0h		R/W-1h	R/W-1h	R/W-1h	R/W-0h	R/W-0h	R/W-0h

**Table 3-50. PCLKCR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R-0	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-20	RESERVED	R-0	0h	Reserved
19	RESERVED	R/W	0h	Reserved
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 Reset type: SYSRSn
17	RESERVED	R-0	0h	Reserved
16	HRCAL	R/W	0h	HRCAL Clock Enable Bit: When set, this enables the clock to the HRCAL module 1: HRCAL clock is enabled 0: HRCAL clock is disabled Reset type: SYSRSn
15	RESERVED	R-0	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12-6	RESERVED	R-0	0h	Reserved
5	CPUTIMER2	R/W	1h	CPUTIMER2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	CPUTIMER1	R/W	1h	CPUTIMER1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-50. 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 Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

### 3.15.4.5 PCLKCR2 Register (Offset = 26h) [Reset = 0000000h]

PCLKCR2 is shown in [Figure 3-43](#) and described in [Table 3-51](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - ETPWM

**Figure 3-43. 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							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
R/W-0h							

**Table 3-51. PCLKCR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	EPWM7	R/W	0h	EPWM7 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
5	EPWM6	R/W	0h	EPWM6 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
4	EPWM5	R/W	0h	EPWM5 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
3	EPWM4	R/W	0h	EPWM4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

**Table 3-51. PCLKCR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	EPWM3	R/W	0h	EPWM3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	EPWM2	R/W	0h	EPWM2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	EPWM1	R/W	0h	EPWM1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.6 PCLKCR3 Register (Offset = 28h) [Reset = 0000000h]

PCLKCR3 is shown in [Figure 3-44](#) and described in [Table 3-52](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - ECAP

**Figure 3-44. 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	RESERVED	RESERVED	RESERVED	RESERVED	ECAP2	ECAP1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-52. PCLKCR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	ECAP2	R/W	0h	ECAP2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	ECAP1	R/W	0h	ECAP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.7 PCLKCR4 Register (Offset = 2Ah) [Reset = 0000000h]

PCLKCR4 is shown in [Figure 3-45](#) and described in [Table 3-53](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - EQEP

**Figure 3-45. PCLKCR4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	RESERVED	RESERVED	EQEP1
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-53. PCLKCR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	EQEP1	R/W	0h	EQEP1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.8 PCLKCR7 Register (Offset = 30h) [Reset = 0000000h]

PCLKCR7 is shown in [Figure 3-46](#) and described in [Table 3-54](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - SCI

**Figure 3-46. PCLKCR7 Register**

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

**Table 3-54. PCLKCR7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	SCI_C	R/W	0h	SCI_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	SCI_B	R/W	0h	SCI_B Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	SCI_A	R/W	0h	SCI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.9 PCLKCR8 Register (Offset = 32h) [Reset = 0000000h]

PCLKCR8 is shown in [Figure 3-47](#) and described in [Table 3-55](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - SPI

**Figure 3-47. 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/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	SPI_A
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-55. PCLKCR8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	SPI_A	R/W	0h	SPI_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.10 PCLKCR9 Register (Offset = 34h) [Reset = 0000000h]

PCLKCR9 is shown in [Figure 3-48](#) and described in [Table 3-56](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - I2C

**Figure 3-48. PCLKCR9 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R-0-0h						R/W-0h	R/W-0h

**Table 3-56. PCLKCR9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	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 Reset type: SYSRSn
0	I2C_A	R/W	0h	I2C_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.11 PCLKCR10 Register (Offset = 36h) [Reset = 0000000h]

PCLKCR10 is shown in [Figure 3-49](#) and described in [Table 3-57](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - CAN

**Figure 3-49. 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	CAN_A						
R/W-0h	R/W-0h						

**Table 3-57. PCLKCR10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	CAN_A	R/W	0h	CAN_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.12 PCLKCR13 Register (Offset = 3Ch) [Reset = 0000000h]

PCLKCR13 is shown in [Figure 3-50](#) and described in [Table 3-58](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - ADC

**Figure 3-50. PCLKCR13 Register**

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

**Table 3-58. PCLKCR13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	ADC_C	R/W	0h	ADC_C Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	ADC_A	R/W	0h	ADC_A Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.13 PCLKCR14 Register (Offset = 3Eh) [Reset = 0000000h]

PCLKCR14 is shown in [Figure 3-51](#) and described in [Table 3-59](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - CMPSS

**Figure 3-51. PCLKCR14 Register**

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

**Table 3-59. PCLKCR14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	CMPSS4	R/W	0h	CMPSS4 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
2	CMPSS3	R/W	0h	CMPSS3 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
1	CMPSS2	R/W	0h	CMPSS2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn
0	CMPSS1	R/W	0h	CMPSS1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.14 PCLKCR21 Register (Offset = 4Ch) [Reset = 0000000h]

PCLKCR21 is shown in [Figure 3-52](#) and described in [Table 3-60](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - DCC

**Figure 3-52. PCLKCR21 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						RESERVED	DCC0
R-0-0h						R/W-0h	R/W-0h

**Table 3-60. PCLKCR21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	DCC0	R/W	0h	DCC Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.15 PCLKCR27 Register (Offset = 58h) [Reset = 0000000h]

PCLKCR27 is shown in [Figure 3-53](#) and described in [Table 3-61](#).

Return to the [Summary Table](#).

Peripheral Clock Gating Register - EPG

**Figure 3-53. PCLKCR27 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							EPG1
R-0-0h							R/W-0h

**Table 3-61. PCLKCR27 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	EPG1	R/W	0h	EPG1 Clock Enable Bit: 0: Module clock is gated-off 1: Module clock is turned-on Reset type: SYSRSn

### 3.15.4.16 SIMRESET Register (Offset = 70h) [Reset = 0000000h]

SIMRESET is shown in [Figure 3-54](#) and described in [Table 3-62](#).

Return to the [Summary Table](#).

Simulated Reset Register

Note: This register exists only on CPU1

**Figure 3-54. SIMRESET 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						XRSn	CPU1RSn
R-0-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 3-62. SIMRESET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Write to this register succeeds only if this field is written with a value of 0xa5a5 Note: [1] Due to this KEY, only 32-bit writes will succeed (provided the KEY matches). 16-bit writes to the upper or lower half of this register will be ignored Reset type: XRSn
15-2	RESERVED	R-0	0h	Reserved
1	XRSn	R-0/W1S	0h	Writing a 1 to this field generates a XRSn like reset. Writing a 0 has no effect. Note: Writing to this pin will pull the XRSn pin low for 512 INTOSC1 clock cycles. Reset type: XRSn
0	CPU1RSn	R-0/W1S	0h	Writing a 1 to this field generates a reset to to CPU1. Writing a 0 has no effect. Reset type: XRSn

### 3.15.4.17 LPMCR Register (Offset = 76h) [Reset = 00000FCh]

LPMCR is shown in [Figure 3-55](#) and described in [Table 3-63](#).

Return to the [Summary Table](#).

LPM Control Register

**Figure 3-55. LPMCR Register**

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

**Table 3-63. LPMCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W1S	0h	Reserved
30-18	RESERVED	R-0	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15	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 Reset type: SYSRSn
14-8	RESERVED	R-0	0h	Reserved
7-2	QUALSTDBY	R/W	3Fh	Select number of OSCCLK clock cycles to qualify the selected inputs when waking the from STANDBY mode: 000000 = 2 OSCCLKs 000001 = 3 OSCCLKs ..... 111111 = 65 OSCCLKs Note: The LPMCR.QUALSTDBY register should be set to a value greater than the ratio of INTOSC1/PLLSYSCLK to ensure proper wake up. Reset type: SYSRSn
1-0	LPM	R/W	0h	These bits set the low power mode for the device. Takes effect when CPU executes the IDLE instruction (when IDLE instruction is out of EXE Phase of the Pipeline) 00: IDLE Mode 01: STANDBY Mode 1x: HALT Mode Reset type: SYSRSn

### 3.15.4.18 GPIOLPMSEL0 Register (Offset = 78h) [Reset = 0000000h]

GPIOLPMSEL0 is shown in [Figure 3-56](#) and described in [Table 3-64](#).

Return to the [Summary Table](#).

GPIO LPM Wakeup select registers

Connects the selected pin to the LPM circuit. Refer to LPM section of the TRM for the wakeup capabilities of the selected pin.

**Figure 3-56. GPIOLPMSEL0 Register**

31	30	29	28	27	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0h							
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0h							
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h							
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h							

**Table 3-64. GPIOLPMSEL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO31	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
30	GPIO30	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
29	GPIO29	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
28	GPIO28	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
27	GPIO27	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
26	GPIO26	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
25	GPIO25	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
24	GPIO24	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
23	GPIO23	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-64. GPIO\_LPMSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	GPIO22	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
21	GPIO21	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
20	GPIO20	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
19	GPIO19	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
18	GPIO18	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
17	GPIO17	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
16	GPIO16	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
15	GPIO15	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
14	GPIO14	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
13	GPIO13	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
12	GPIO12	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
11	GPIO11	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
10	GPIO10	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
9	GPIO9	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
8	GPIO8	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
7	GPIO7	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
6	GPIO6	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-64. GPIOLPMSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	GPIO5	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
4	GPIO4	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
3	GPIO3	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
2	GPIO2	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
1	GPIO1	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
0	GPIO0	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

### 3.15.4.19 GPIO\_LPMSEL1 Register (Offset = 7Ah) [Reset = 0000000h]

GPIO\_LPMSEL1 is shown in [Figure 3-57](#) and described in [Table 3-65](#).

Return to the [Summary Table](#).

GPIO LPM Wakeup select registers

Connects the selected pin to the LPM circuit. Refer to LPM section of the TRM for the wakeup capabilities of the selected pin.

**Figure 3-57. GPIO\_LPMSEL1 Register**

31	30	29	28	27	26	25	24
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0h							
23	22	21	20	19	18	17	16
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0h							
15	14	13	12	11	10	9	8
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0h							

**Table 3-65. GPIO\_LPMSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	GPIO63	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
30	GPIO62	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
29	GPIO61	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
28	GPIO60	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
27	GPIO59	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
26	GPIO58	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
25	GPIO57	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
24	GPIO56	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
23	GPIO55	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-65. GPIO\_LPMSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	GPIO54	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
21	GPIO53	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
20	GPIO52	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
19	GPIO51	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
18	GPIO50	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
17	GPIO49	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
16	GPIO48	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
15	GPIO47	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
14	GPIO46	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
13	GPIO45	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
12	GPIO44	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
11	GPIO43	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
10	GPIO42	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
9	GPIO41	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
8	GPIO40	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
7	GPIO39	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
6	GPIO38	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

**Table 3-65. GPIOLPMSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	GPIO37	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
4	GPIO36	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
3	GPIO35	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
2	GPIO34	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
1	GPIO33	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn
0	GPIO32	R/W	0h	0 pin is dis-connected from LPM circuit 1 pin is connected to LPM circuit Reset type: SYSRSn

### 3.15.4.20 TMR2CLKCTL Register (Offset = 7Ch) [Reset = 0000000h]

TMR2CLKCTL is shown in [Figure 3-58](#) and described in [Table 3-66](#).

Return to the [Summary Table](#).

Timer2 Clock Measurement functionality control register

This memory mapped register requires a delay of 45 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-58. TMR2CLKCTL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED		TMR2CLKPRESCALE			TMR2CLKSRCSEL		
R-0-0h		R/W-0h			R/W-0h		

**Table 3-66. TMR2CLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-3	TMR2CLKPRESCALE	R/W	0h	CPU Timer 2 Clock Pre-Scale Value: These bits select the pre-scale value for the selected clock source for CPU Timer 2: 0,0,0,/1 (default on reset) 0,0,1,/2, 0,1,0,/4 0,1,1,/8 1,0,0,/16 1,0,1,spare (defaults to /16) 1,1,0,spare (defaults to /16) 1,1,1,spare (defaults to /16) Note: [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. Reset type: SYSRSn
2-0	TMR2CLKSRCSEL	R/W	0h	CPU Timer 2 Clock Source Select Bit: This bit selects the source for CPU Timer 2: 000 =SYSCLK Selected (default on reset, pre-scale is bypassed) 001 = INTOSC1 010 = INTOSC2 011 = XTAL 100 = PUMPOSC (from no-wrapper) 101 = FOSCCLK (Reserved) 110 = AUXPLLCLK (Reserved) 111 = reserved Reset type: SYSRSn

### 3.15.4.21 RESCCLR Register (Offset = 7Eh) [Reset = 0000000h]

RESCCLR is shown in [Figure 3-59](#) and described in [Table 3-67](#).

Return to the [Summary Table](#).

Reset Cause Clear Register

**Figure 3-59. RESCCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED				SIMRESET_XR Sn	SIMRESET_CP U1RSn	RESERVED	SCCRESETn
R-0-0h				W1C-0h	W1C-0h	R-0-0h	W1S-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	NMIWDRSn	WDRSn	XRSn	POR
R-0-0h	W1S-0h	W1S-0h	R-0-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h

**Table 3-67. RESCCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-12	RESERVED	R-0	0h	Reserved
11	SIMRESET_XRSn	W1C	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
10	SIMRESET_CPU1RSn	W1C	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
9	RESERVED	R-0	0h	Reserved
8	SCCRESETn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	RESERVED	W1S	0h	Reserved
5	RESERVED	W1S	0h	Reserved
4	RESERVED	R-0	0h	Reserved
3	NMIWDRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn

**Table 3-67. RESCCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	WDRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
1	XRSn	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn
0	POR	W1S	0h	Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0. Writing a 1 to this bit clears the status bit in RESC to 0 Writing 0 has no effect. Reset type: SYSRSn

### 3.15.4.22 RESC Register (Offset = 80h) [Reset = X000003h]

RESC is shown in [Figure 3-60](#) and described in [Table 3-68](#).

Return to the [Summary Table](#).

Reset Cause register

**Figure 3-60. RESC Register**

31	30	29	28	27	26	25	24
DCON	XRSn_pin_status	RESERVED					
R-0h	R-X	R-0-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED				SIMRESET_XRSn	SIMRESET_CPU1RSn	RESERVED	SCCRESETn
R-0-0h				R-0h	R-0h	R-0-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	NMIWDRSn	WDRSn	XRSn	POR
R-0-0h	R-0h	R-0h	R-0-0h	R-0h	R-0h	R-1h	R-1h

**Table 3-68. RESC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	DCON	R	0h	Reading this bit provides the status of debugger connection to the C28x CPU. 0 : Debugger is not connected to the C28x CPU 1 : Debugger is connected to the C28x CPU Notes: [1] This bit is connected to the DCON o/p signal of the C28x CPU Reset type: N/A
30	XRSn_pin_status	R	X	Reading this bit provides the current status of the XRSn pin. Reset value is reflective of the pin status. Reset type: N/A
29-16	RESERVED	R-0	0h	Reserved
15-12	RESERVED	R-0	0h	Reserved
11	SIMRESET_XRSn	R	0h	If this bit is set, indicates that the device was reset by SIMRESET_XRSn Reset type: PORESETn
10	SIMRESET_CPU1RSn	R	0h	If this bit is set, indicates that the device was reset by SIMRESET_CPU1RSn Reset type: PORESETn
9	RESERVED	R-0	0h	Reserved
8	SCCRESETn	R	0h	If this bit is set, indicates that the device was reset by SCCRESETn (fired by DCISM). Reset type: PORESETn
7	RESERVED	R-0	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R-0	0h	Reserved

**Table 3-68. RESC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	NMIWDRSn	R	0h	If this bit is set, indicates that the device was reset by NMIWDRSn. Note: To know the exact cause of NMI after the reset, software needs to read NMISHDFLG registers Reset type: PORESETn
2	WDRSn	R	0h	If this bit is set, indicates that the device was reset by WDRSn. Note: [1] A bit inside WD module also provides the same information. This bit is present to keep things consistent. This register is a one-stop shop for the software to know the reset cause for the C28x core. Reset type: PORESETn
1	XRSn	R	1h	If this bit is set, indicates that the device was reset by XRSn. Reset type: PORESETn
0	POR	R	1h	If this bit is set, indicates that the device was reset by PORn. Reset type: PORESETn

### 3.15.4.23 USER\_REG1\_SYSRSn Register (Offset = A0h) [Reset = 0000000h]

USER\_REG1\_SYSRSn is shown in [Figure 3-61](#) and described in [Table 3-69](#).

Return to the [Summary Table](#).

Software Configurable registers reset by SYSRSn

**Figure 3-61. USER\_REG1\_SYSRSn Register**

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

**Table 3-69. USER\_REG1\_SYSRSn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by SYSRSn to be used by the application software Reset type: SYSRSn

### 3.15.4.24 USER\_REG2\_SYSRSn Register (Offset = A2h) [Reset = 0000000h]

USER\_REG2\_SYSRSn is shown in [Figure 3-62](#) and described in [Table 3-70](#).

Return to the [Summary Table](#).

Software Configurable registers reset by SYSRSn

**Figure 3-62. USER\_REG2\_SYSRSn Register**

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

**Table 3-70. USER\_REG2\_SYSRSn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by SYSRSn to be used by the application software Reset type: SYSRSn

### 3.15.4.25 USER\_REG1\_XRSn Register (Offset = A4h) [Reset = 00000000h]

USER\_REG1\_XRSn is shown in [Figure 3-63](#) and described in [Table 3-71](#).

Return to the [Summary Table](#).

Software Configurable registers reset by XRSn

**Figure 3-63. USER\_REG1\_XRSn Register**

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

**Table 3-71. USER\_REG1\_XRSn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by XRSn to be used by the application software Reset type: XRSn

### 3.15.4.26 USER\_REG2\_XRSn Register (Offset = A6h) [Reset = 00000000h]

USER\_REG2\_XRSn is shown in [Figure 3-64](#) and described in [Table 3-72](#).

Return to the [Summary Table](#).

Software Configurable registers reset by XRSn

**Figure 3-64. USER\_REG2\_XRSn Register**

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

**Table 3-72. USER\_REG2\_XRSn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by XRSn to be used by the application software Reset type: XRSn

### 3.15.4.27 USER\_REG1\_PORESETn Register (Offset = A8h) [Reset = 0000000h]

USER\_REG1\_PORESETn is shown in [Figure 3-65](#) and described in [Table 3-73](#).

Return to the [Summary Table](#).

Software Configurable registers reset by PORESETn

**Figure 3-65. USER\_REG1\_PORESETn Register**

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

**Table 3-73. USER\_REG1\_PORESETn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by PORESETn to be used by the application software Reset type: PORESETn

### 3.15.4.28 USER\_REG2\_PORESETn Register (Offset = AAh) [Reset = 0000000h]

USER\_REG2\_PORESETn is shown in [Figure 3-66](#) and described in [Table 3-74](#).

Return to the [Summary Table](#).

Software Configurable registers reset by PORESETn

**Figure 3-66. USER\_REG2\_PORESETn Register**

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

**Table 3-74. USER\_REG2\_PORESETn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by PORESETn to be used by the application software Reset type: PORESETn

### 3.15.4.29 USER\_REG3\_PORESETn Register (Offset = ACh) [Reset = 0000000h]

USER\_REG3\_PORESETn is shown in [Figure 3-67](#) and described in [Table 3-75](#).

Return to the [Summary Table](#).

Software Configurable registers reset by PORESETn

**Figure 3-67. USER\_REG3\_PORESETn Register**

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

**Table 3-75. USER\_REG3\_PORESETn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by PORESETn to be used by the application software Reset type: PORESETn

### 3.15.4.30 USER\_REG4\_PORESETn Register (Offset = AEh) [Reset = 0000000h]

USER\_REG4\_PORESETn is shown in [Figure 3-68](#) and described in [Table 3-76](#).

Return to the [Summary Table](#).

Software Configurable registers reset by PORESETn

**Figure 3-68. USER\_REG4\_PORESETn Register**

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

**Table 3-76. USER\_REG4\_PORESETn Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BITS	R/W	0h	R/W bits reset by PORESETn to be used by the application software Reset type: PORESETn

### 3.15.5 CPUTIMER\_REGS Registers

Table 3-77 lists the memory-mapped registers for the CPUTIMER\_REGS registers. All register offset addresses not listed in Table 3-77 should be considered as reserved locations and the register contents should not be modified.

**Table 3-77. CPUTIMER\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TIM	CPU-Timer, Counter Register		<a href="#">Go</a>
2h	PRD	CPU-Timer, Period Register		<a href="#">Go</a>
4h	TCR	CPU-Timer, Control Register		<a href="#">Go</a>
6h	TPR	CPU-Timer, Prescale Register		<a href="#">Go</a>
7h	TPRH	CPU-Timer, Prescale Register High		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-78 shows the codes that are used for access types in this section.

**Table 3-78. CPUTIMER\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

### 3.15.5.1 TIM Register (Offset = 0h) [Reset = 0000FFFFh]

TIM is shown in [Figure 3-69](#) and described in [Table 3-79](#).

Return to the [Summary Table](#).

CPU-Timer, Counter Register

**Figure 3-69. TIM Register**

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

**Table 3-79. TIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	0h	CPU-Timer Counter Registers The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (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. Reset type: SYSRSn
15-0	LSW	R/W	FFFFh	CPU-Timer Counter Registers The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (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. Reset type: SYSRSn

### 3.15.5.2 PRD Register (Offset = 2h) [Reset = 0000FFFFh]

PRD is shown in [Figure 3-70](#) and described in [Table 3-80](#).

Return to the [Summary Table](#).

CPU-Timer, Period Register

**Figure 3-70. 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-0h																R/W-FFFFh															

**Table 3-80. PRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	MSW	R/W	0h	CPU-Timer Period Registers The PRDH register holds the high 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR). Reset type: SYSRSn
15-0	LSW	R/W	FFFFh	CPU-Timer Period Registers The PRD register holds the low 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR). Reset type: SYSRSn

### 3.15.5.3 TCR Register (Offset = 4h) [Reset = 0001h]

TCR is shown in [Figure 3-71](#) and described in [Table 3-81](#).

Return to the [Summary Table](#).

CPU-Timer, Control Register

**Figure 3-71. TCR Register**

15		14		13		12		11		10		9		8	
TIF		TIE		RESERVED				FREE		SOFT		RESERVED			
R/W1C-0h		R/W-0h		R-0h				R/W-0h		R/W-0h		R-0h			
7		6		5		4		3		2		1		0	
RESERVED				TRB		TSS		RESERVED							
R-0h				R/W-0h		R/W-0h		R-1h							

**Table 3-81. TCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	TIF	R/W1C	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. Reset type: SYSRSn 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. Reset type: SYSRSn 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. Reset type: SYSRSn 0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop) (SOFT bit controls the emulation behavior) 1h (R/W) = Free Run (SOFT bit is don't care, counter is free running)
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. Reset type: SYSRSn 0h (R/W) = Stop after the next decrement of the TIMH:TIM (hard stop). (ONLY if FREE=0, if FREE=1 this bit is don't care) 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). (ONLY if FREE=0, if FREE=1 this bit is don't care)
9-6	RESERVED	R	0h	Reserved

**Table 3-81. TCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	TRB	R/W	0h	Timer reload Reset type: SYSRSn 0h (R/W) = The TRB bit is always read as zero. Writes of 0 are ignored. 1h (R/W) = When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD, and the prescaler counter (PSCH:PSC) is loaded with the value in the timer dividedown register (TDDR:H:TDDR).
4	TSS	R/W	0h	CPU-Timer stop status bit. TSS is a 1-bit flag that stops or starts the CPU-timer. Reset type: SYSRSn 0h (R/W) = Reads of 0 indicate the CPU-timer is running. To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the CPU-timer immediately starts. 1h (R/W) = Reads of 1 indicate that the CPU-timer is stopped. To stop the CPU-timer, set TSS to 1.
3-0	RESERVED	R	1h	Reserved

### 3.15.5.4 TPR Register (Offset = 6h) [Reset = 0000h]

TPR is shown in [Figure 3-72](#) and described in [Table 3-82](#).

Return to the [Summary Table](#).

CPU-Timer, Prescale Register

**Figure 3-72. TPR Register**

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

**Table 3-82. TPR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	PSC	R	0h	CPU-Timer Prescale Counter. 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. Reset type: SYSRSn
7-0	TDDR	R/W	0h	CPU-Timer Divide-Down. 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. Reset type: SYSRSn

### 3.15.5.5 TPRH Register (Offset = 7h) [Reset = 0000h]

TPRH is shown in [Figure 3-73](#) and described in [Table 3-83](#).

Return to the [Summary Table](#).

CPU-Timer, Prescale Register High

**Figure 3-73. TPRH Register**

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

**Table 3-83. TPRH Register Field Descriptions**

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

### 3.15.6 DEV\_CFG\_REGS Registers

Table 3-84 lists the memory-mapped registers for the DEV\_CFG\_REGS registers. All register offset addresses not listed in Table 3-84 should be considered as reserved locations and the register contents should not be modified.

**Table 3-84. DEV\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
8h	PARTIDL	Lower 32-bit of Device PART Identification Number		<a href="#">Go</a>
Ah	PARTIDH	Upper 32-bit of Device PART Identification Number		<a href="#">Go</a>
Ch	REVID	Device Revision Number		<a href="#">Go</a>
74h	TRIMERRSTS	TRIM Error Status register		<a href="#">Go</a>
86h	SOFTPRES2	EPWM Software Reset register	EALLOW	<a href="#">Go</a>
88h	SOFTPRES3	ECAP Software Reset register	EALLOW	<a href="#">Go</a>
8Ah	SOFTPRES4	EQEP Software Reset register	EALLOW	<a href="#">Go</a>
90h	SOFTPRES7	SCI Software Reset register	EALLOW	<a href="#">Go</a>
92h	SOFTPRES8	SPI Software Reset register	EALLOW	<a href="#">Go</a>
94h	SOFTPRES9	I2C Software Reset register	EALLOW	<a href="#">Go</a>
96h	SOFTPRES10	CAN Software Reset register	EALLOW	<a href="#">Go</a>
9Ch	SOFTPRES13	ADC Software Reset register	EALLOW	<a href="#">Go</a>
9Eh	SOFTPRES14	CMPSS Software Reset register	EALLOW	<a href="#">Go</a>
ACh	SOFTPRES21	DCC Software Reset register	EALLOW	<a href="#">Go</a>
B8h	SOFTPRES27	EPG Software Reset register	EALLOW	<a href="#">Go</a>
BAh	SOFTPRES28	Flash Software Reset register	EALLOW	<a href="#">Go</a>
130h	TAP_STATUS	Status of JTAG State machine & Debugger Connect		<a href="#">Go</a>
19Bh	ECAPTYPE	Configures ECAP Type for the device	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-85 shows the codes that are used for access types in this section.

**Table 3-85. DEV\_CFG\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
WSonce	WSonce	Write Set once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		

**Table 3-85. DEV\_CFG\_REGS Access Type Codes (continued)**

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

### 3.15.6.1 PARTIDL Register (Offset = 8h) [Reset = 00XXXX0h]

PARTIDL is shown in [Figure 3-74](#) and described in [Table 3-86](#).

Return to the [Summary Table](#).

Lower 32-bit of Device PART Identification Number

**Figure 3-74. PARTIDL Register**

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

**Table 3-86. PARTIDL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	RESERVED	R	0h	Reserved
23-16	RESERVED	R	X	Reserved
15	RESERVED	R	0h	Reserved
14-13	INSTASPIN	R	X	1 = InstaSPIN-FOC 2 = NONE 3 = NONE Reset type: PORESETn
12	RESERVED	R	0h	Reserved
11	RESERVED	R	X	Reserved
10-8	PIN_COUNT	R	X	0 = 56 pin QFN 1 = 64 pin (QFP - Q100) 2 = 64 pin (QFP) 3 = 80 pin (QFP) 4 = 48 pin (QFP) 5 = 32 pin (QFN) 6 = Reserved 7 = 48 pin (QFN) 8 = 64 pin (QFP) - VREGENZ bondout Reset type: PORESETn
7-6	QUAL	R	X	0 = Engineering sample (TMX) 1 = Pilot production (TMP) 2 = Fully qualified (TMS) Reset type: PORESETn
5	RESERVED	R	0h	Reserved
4-3	RESERVED	R	0h	Reserved
2-0	RESERVED	R	0h	Reserved

### 3.15.6.2 PARTIDH Register (Offset = Ah) [Reset = 06XX0500h]

PARTIDH is shown in [Figure 3-75](#) and described in [Table 3-87](#).

Return to the [Summary Table](#).

Upper 32-bit of Device PART Identification Number

**Figure 3-75. PARTIDH Register**

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

**Table 3-87. PARTIDH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	DEVICE_CLASS_ID	R	6h	Device class ID Reset type: PORESETn
23-16	PARTNO	R	X	Refer to Datasheet for Device Part Number Reset type: PORESETn
15-8	FAMILY	R	5h	Device Family Reset type: PORESETn
7-4	RESERVED	R	0h	Reserved
3-0	RESERVED	R	0h	Reserved

### 3.15.6.3 REVID Register (Offset = Ch) [Reset = 00000000h]

REVID is shown in [Figure 3-76](#) and described in [Table 3-88](#).

Return to the [Summary Table](#).

Device Revision Number

**Figure 3-76. REVID Register**

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

**Table 3-88. REVID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-0	REVID	R/WOnce	0h	Device Revision ID. Loaded from flash trim sector by boot rom. Reset value is die-specific. Reset type: XRSn

### 3.15.6.4 TRIMERRSTS Register (Offset = 74h) [Reset = 0000000h]

TRIMERRSTS is shown in [Figure 3-77](#) and described in [Table 3-89](#).

Return to the [Summary Table](#).

TRIM Error Status register

**Figure 3-77. TRIMERRSTS Register**

31	30	29	28	27	26	25	24	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																LERR															
R-0-0h																R/WSoOnce-0h															

**Table 3-89. TRIMERRSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-0	LERR	R/WSoOnce	0h	TRIM information load error status. This will include error during SRAM repair also. 0x1: Correctable single bit error 0x2: Uncorrectable double bit error 0x20: Trim over timeout error Other: Non zero value indicates error during load Note: [1] This bit is updated by software. Details will be filled in once the Boot ROM related requirements are complete. It should have bits to indicate (i) Double bit error during trim load (ii) Single bit error during trim load (iii) Double bit error during SRAM repair load (iv) Single bit error error during SRAM repair load (v) SRAM repair error load (chain is broken) (vi) PWRUPSTS.TRIMOVER signal is not asserted even after the full wait time Reset type: XRSn

### 3.15.6.5 SOFTPRES2 Register (Offset = 86h) [Reset = 0000000h]

SOFTPRES2 is shown in [Figure 3-78](#) and described in [Table 3-90](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-78. 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							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	EPWM7	EPWM6	EPWM5	EPWM4	EPWM3	EPWM2	EPWM1
R/W-0h							

**Table 3-90. SOFTPRES2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	EPWM7	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
5	EPWM6	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
4	EPWM5	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
3	EPWM4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	EPWM3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

**Table 3-90. 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 Reset type: SYSRSn
0	EPWM1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.6 SOFTPRES3 Register (Offset = 88h) [Reset = 0000000h]

SOFTPRES3 is shown in [Figure 3-79](#) and described in [Table 3-91](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-79. 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	RESERVED	RESERVED	RESERVED	RESERVED	ECAP2	ECAP1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-91. SOFTPRES3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	ECAP2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	ECAP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.7 SOFTPRES4 Register (Offset = 8Ah) [Reset = 0000000h]

SOFTPRES4 is shown in [Figure 3-80](#) and described in [Table 3-92](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-80. SOFTPRES4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				RESERVED	RESERVED	RESERVED	EQEP1
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-92. SOFTPRES4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	EQEP1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.8 SOFTPRES7 Register (Offset = 90h) [Reset = 0000000h]

SOFTPRES7 is shown in [Figure 3-81](#) and described in [Table 3-93](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-81. SOFTPRES7 Register**

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

**Table 3-93. SOFTPRES7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	SCI_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	SCI_B	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	SCI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.9 SOFTPRES8 Register (Offset = 92h) [Reset = 0000000h]

SOFTPRES8 is shown in [Figure 3-82](#) and described in [Table 3-94](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-82. 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/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	SPI_A
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-94. SOFTPRES8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-18	RESERVED	R-0	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	SPI_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.10 SOFTPRES9 Register (Offset = 94h) [Reset = 0000000h]

SOFTPRES9 is shown in [Figure 3-83](#) and described in [Table 3-95](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-83. SOFTPRES9 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						I2C_B	I2C_A
R-0-0h						R/W-0h	R/W-0h

**Table 3-95. SOFTPRES9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	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 Reset type: SYSRSn
0	I2C_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.11 SOFTPRES10 Register (Offset = 96h) [Reset = 00000X0h]

SOFTPRES10 is shown in [Figure 3-84](#) and described in [Table 3-96](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-84. SOFTPRES10 Register**

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

**Table 3-96. SOFTPRES10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R	X	Reserved
6	RESERVED	R	X	Reserved
5	RESERVED	R	X	Reserved
4	RESERVED	R	X	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	CAN_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.12 SOFTPRES13 Register (Offset = 9Ch) [Reset = 0000000h]

SOFTPRES13 is shown in [Figure 3-85](#) and described in [Table 3-97](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-85. SOFTPRES13 Register**

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

**Table 3-97. SOFTPRES13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	ADC_C	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	ADC_A	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.13 SOFTPRES14 Register (Offset = 9Eh) [Reset = 0000000h]

SOFTPRES14 is shown in [Figure 3-86](#) and described in [Table 3-98](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-86. SOFTPRES14 Register**

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

**Table 3-98. SOFTPRES14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	CMPSS4	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
2	CMPSS3	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
1	CMPSS2	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn
0	CMPSS1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.14 SOFTPRES21 Register (Offset = ACh) [Reset = 0000000h]

SOFTPRES21 is shown in [Figure 3-87](#) and described in [Table 3-99](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-87. SOFTPRES21 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						RESERVED	DCC0
R-0-0h						R/W-0h	R/W-0h

**Table 3-99. SOFTPRES21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	DCC0	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.15 SOFTPRES27 Register (Offset = B8h) [Reset = 0000000h]

SOFTPRES27 is shown in [Figure 3-88](#) and described in [Table 3-100](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-88. SOFTPRES27 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							EPG1
R-0-0h							R/W-0h

**Table 3-100. SOFTPRES27 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	EPG1	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Reset type: SYSRSn

### 3.15.6.16 SOFTPRES28 Register (Offset = BAh) [Reset = 0000000h]

SOFTPRES28 is shown in [Figure 3-89](#) and described in [Table 3-101](#).

Return to the [Summary Table](#).

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-89. SOFTPRES28 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							FLASHA
R-0-0h							R/W-0h

**Table 3-101. SOFTPRES28 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	FLASHA	R/W	0h	1: Module is under reset 0: Module reset is determined by the normal device reset structure Note: Whenever the reset to flash is asserted, it will be internally stretched to ~15us Reset type: SYSRSn

### 3.15.6.17 TAP\_STATUS Register (Offset = 130h) [Reset = 0000000h]

TAP\_STATUS is shown in [Figure 3-90](#) and described in [Table 3-102](#).

Return to the [Summary Table](#).

Status of JTAG State machine & Debugger Connect

**Figure 3-90. TAP\_STATUS Register**

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

**Table 3-102. TAP\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	DCON	R	0h	DebugConnect indication from IcePick. Reset type: PORESETn
30-16	RESERVED	R-0	0h	Reserved
15-0	TAP_STATE	R	0h	TAP State Vector. With bits representing, Connect coresponding POTAP* output to the 0:TLR, 1:IDLE, 2:SELECTDR, 3:CAPDR, 4:SHIFDR, 5:EXIT1DR, 6:PAUSEDR, 7:EXIT2DR, 8:UPDR, 9:SELECTIR, 10:CAPIR, 11:SHIFIR, 12:EXIT1IR, 13:PAUSEIR, 14:EXIT2IR, 15:UPDIR, Reset type: PORESETn

### 3.15.6.18 ECAPTYPE Register (Offset = 19Bh) [Reset = 0000h]

ECAPTYPE is shown in [Figure 3-91](#) and described in [Table 3-103](#).

Return to the [Summary Table](#).

Based on the configuration enables/disables features associated with the ECAP type.

**Figure 3-91. ECAPTYPE Register**

15	14	13	12	11	10	9	8
LOCK		RESERVED					
R/WOnce-0h		R-0-0h					
7	6	5	4	3	2	1	0
RESERVED						TYPE	
R-0-0h						R/W-0h	

**Table 3-103. ECAPTYPE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	LOCK	R/WOnce	0h	1: Write to this register is not allowed. 0: Write to this register is allowed. Reset type: SYSRSn
14-2	RESERVED	R-0	0h	Reserved
1-0	TYPE	R/W	0h	'00,10,11': 1. No EALLOW protection to ECAP registers. '01': 1. ECAP registers are EALLOW protected. Reset type: SYSRSn

### 3.15.7 MEM\_CFG\_REGS Registers

Table 3-104 lists the memory-mapped registers for the MEM\_CFG\_REGS registers. All register offset addresses not listed in Table 3-104 should be considered as reserved locations and the register contents should not be modified.

**Table 3-104. MEM\_CFG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DxLOCK	Dedicated RAM Config Lock Register	EALLOW	<a href="#">Go</a>
2h	DxCOMMIT	Dedicated RAM Config Lock Commit Register	EALLOW	<a href="#">Go</a>
8h	DxACCPROT0	Dedicated RAM Config Register	EALLOW	<a href="#">Go</a>
Ah	DxACCPROT1	Dedicated RAM Config Register	EALLOW	<a href="#">Go</a>
10h	DxTEST	Dedicated RAM TEST Register		<a href="#">Go</a>
12h	DxINIT	Dedicated RAM Init Register	EALLOW	<a href="#">Go</a>
14h	DxINITDONE	Dedicated RAM InitDone Status Register		<a href="#">Go</a>
16h	DxRAMTEST_LOCK	Lock register to Dx RAM TEST registers		<a href="#">Go</a>
20h	LSxLOCK	Local Shared RAM Config Lock Register	EALLOW	<a href="#">Go</a>
22h	LSxCOMMIT	Local Shared RAM Config Lock Commit Register	EALLOW	<a href="#">Go</a>
28h	LSxACCPROT0	Local Shared RAM Config Register 0	EALLOW	<a href="#">Go</a>
30h	LSxTEST	Local Shared RAM TEST Register		<a href="#">Go</a>
32h	LSxINIT	Local Shared RAM Init Register	EALLOW	<a href="#">Go</a>
34h	LSxINITDONE	Local Shared RAM InitDone Status Register		<a href="#">Go</a>
36h	LSxRAMTEST_LOCK	Lock register to LSx RAM TEST registers		<a href="#">Go</a>
A0h	ROM_LOCK	ROM Config Lock Register		<a href="#">Go</a>
A2h	ROM_TEST	ROM TEST Register		<a href="#">Go</a>
A4h	ROM_FORCE_ERROR	ROM Force Error register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-105 shows the codes that are used for access types in this section.

**Table 3-105. MEM\_CFG\_REGS Access Type Codes**

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

**Table 3-105. MEM\_CFG\_REGS Access Type Codes (continued)**

Access Type	Code	Description
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 3.15.7.1 DxLOCK Register (Offset = 0h) [Reset = 0000000h]

DxLOCK is shown in [Figure 3-92](#) and described in [Table 3-106](#).

Return to the [Summary Table](#).

Dedicated RAM Config Lock Register

**Figure 3-92. 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_PIEVECT	RESERVED	RESERVED	LOCK_M1	LOCK_M0
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-106. DxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	LOCK_PIEVECT	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for PIEVECT RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed. 1: Write to ACCPROT, INIT and MSEL fields are blocked. Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	LOCK_M1	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for M1 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed. 1: Write to ACCPROT, INIT and MSEL fields are blocked. Reset type: SYSRSn
0	LOCK_M0	R/W	0h	Locks the write to access protection, master select, initialization control and test register fields for M0 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed. 1: Write to ACCPROT, INIT and MSEL fields are blocked. Reset type: SYSRSn

### 3.15.7.2 DxCOMMIT Register (Offset = 2h) [Reset = 0000000h]

DxCOMMIT is shown in [Figure 3-93](#) and described in [Table 3-107](#).

Return to the [Summary Table](#).

Dedicated RAM Config Lock Commit Register

**Figure 3-93. 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_PIEVECT	RESERVED	RESERVED	COMMIT_M1	COMMIT_M0
R-0h			R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 3-107. DxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	COMMIT_PIEVECT	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for PIEVECT RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT, INIT and MSEL fields are permanently blocked. Reset type: SYSRSn
3	RESERVED	R/WOnce	0h	Reserved
2	RESERVED	R/WOnce	0h	Reserved
1	COMMIT_M1	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for M1 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT, INIT and MSEL fields are permanently blocked. Reset type: SYSRSn
0	COMMIT_M0	R/WOnce	0h	Permanently Locks the write to access protection, master select, initialization control and test register fields for M0 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed based on value of lock field in DxLOCK register. 1: Write to ACCPROT, INIT and MSEL fields are permanently blocked. Reset type: SYSRSn

### 3.15.7.3 DxACCPROT0 Register (Offset = 8h) [Reset = 0000000h]

DxACCPROT0 is shown in [Figure 3-94](#) and described in [Table 3-108](#).

Return to the [Summary Table](#).

Dedicated RAM Config Register

**Figure 3-94. DxACCPROT0 Register**

31	30	29	28	27	26	25	24
RESERVED						RESERVED	RESERVED
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						RESERVED	RESERVED
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED						CPUWRPROT_M1	FETCHPROT_M1
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CPUWRPROT_M0	FETCHPROT_M0
R-0h						R/W-0h	R/W-0h

**Table 3-108. DxACCPROT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-18	RESERVED	R	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	CPUWRPROT_M1	R/W	0h	CPU WR Protection For M1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are block. Reset type: SYSRSn
8	FETCHPROT_M1	R/W	0h	Fetch Protection For M1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_M0	R/W	0h	CPU WR Protection For M0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are block. Reset type: SYSRSn
0	FETCHPROT_M0	R/W	0h	Fetch Protection For M0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn

### 3.15.7.4 DxACCPROT1 Register (Offset = Ah) [Reset = 0000000h]

DxACCPROT1 is shown in [Figure 3-95](#) and described in [Table 3-109](#).

Return to the [Summary Table](#).

Dedicated RAM Config Register

**Figure 3-95. DxACCPROT1 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_	RESERVED
R-0h						PIEVECT	R/W-0h
							R/W-0h

**Table 3-109. DxACCPROT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_PIEVECT	R/W	0h	CPU Write Protection For PIEVECT RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
0	RESERVED	R/W	0h	Reserved

### 3.15.7.5 DxTEST Register (Offset = 10h) [Reset = 0000000h]

DxTEST is shown in [Figure 3-96](#) and described in [Table 3-110](#).

Return to the [Summary Table](#).

Dedicated RAM TEST Register

**Figure 3-96. 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						TEST_PIEVECT	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		TEST_M1		TEST_M0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-110. DxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9-8	TEST_PIEVECT	R/W	0h	Selects the different modes for PIEVECT 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. Note: Any non zero value would enable CPU writes over-riding write access protection if any and will not generate a access protection violation. Reset type: SYSRSn
7-6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R/W	0h	Reserved
3-2	TEST_M1	R/W	0h	Selects the different 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: Same as functional mode, but interrupt/NMI is not generated on error. Note: Any non zero value would enable CPU writes over-riding write access protection if any and will not generate a access protection violation. Reset type: SYSRSn
1-0	TEST_M0	R/W	0h	Selects the different 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: Same as functional mode, but interrupt/NMI is not generated on error. Note: Any non zero value would enable CPU writes over-riding write access protection if any and will not generate a access protection violation. Reset type: SYSRSn

### 3.15.7.6 DxINIT Register (Offset = 12h) [Reset = 0000000h]

DxINIT is shown in [Figure 3-97](#) and described in [Table 3-111](#).

Return to the [Summary Table](#).

Dedicated RAM Init Register

**Figure 3-97. 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_PIEVECT	RESERVED	RESERVED	INIT_M1	INIT_M0
R-0h			R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-111. DxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	INIT_PIEVECT	R-0/W1S	0h	RAM Initialization control for PIEVECT RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	INIT_M1	R-0/W1S	0h	RAM Initialization control for M1 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
0	INIT_M0	R-0/W1S	0h	RAM Initialization control for M0 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

### 3.15.7.7 DxINITDONE Register (Offset = 14h) [Reset = 0000000h]

DxINITDONE is shown in [Figure 3-98](#) and described in [Table 3-112](#).

Return to the [Summary Table](#).

Dedicated RAM InitDone Status Register

**Figure 3-98. 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_PIE VECT	RESERVED	RESERVED	INITDONE_M1	INITDONE_M0
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-112. DxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	INITDONE_PIEVECT	R	0h	RAM Initialization status for PIEVECT RAM: 0: RAM Initialization has completed. 1: RAM Initialization has completed. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	INITDONE_M1	R	0h	RAM Initialization status for M1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization has completed. Reset type: SYSRSn
0	INITDONE_M0	R	0h	RAM Initialization status for M0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn

### 3.15.7.8 DxRAMTEST\_LOCK Register (Offset = 16h) [Reset = 0000000h]

DxRAMTEST\_LOCK is shown in [Figure 3-99](#) and described in [Table 3-113](#).

Return to the [Summary Table](#).

Lock register to Dx RAM TEST registers

**Figure 3-99. DxRAMTEST\_LOCK Register**

31	30	29	28	27	26	25	24
KEY							
R-0/W-0h							
23	22	21	20	19	18	17	16
KEY							
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			PIEVECT	RESERVED	RESERVED	M1	M0
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-113. DxRAMTEST\_LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	A value of 0xa5a5 to this field is simultaneously required for the writes to the rest of the fields of this register to succeed, Reset type: SYSRSn
15-5	RESERVED	R	0h	Reserved
4	PIEVECT	R/W	0h	0: Allows writes to DxTEST.TEST_PIEVECT field. 1: Blocks writes to DxTEST.TEST_PIEVECT field Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	M1	R/W	0h	0: Allows writes to DxTEST.TEST_M1 field. 1: Blocks writes to DxTEST.TEST_M1 field Reset type: SYSRSn
0	M0	R/W	0h	0: Allows writes to DxTEST.TEST_M0 field. 1: Blocks writes to DxTEST.TEST_M0 field Reset type: SYSRSn

### 3.15.7.9 LSxLOCK Register (Offset = 20h) [Reset = 0000000h]

LSxLOCK is shown in [Figure 3-100](#) and described in [Table 3-114](#).

Return to the [Summary Table](#).

Local Shared RAM Config Lock Register

**Figure 3-100. 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	LOCK_LS1	LOCK_LS0
R/W-0h							

**Table 3-114. LSxLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	LOCK_LS1	R/W	0h	Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed. 1: Write to ACCPROT, INIT and MSEL fields are blocked. Reset type: SYSRSn
0	LOCK_LS0	R/W	0h	Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed. 1: Write to ACCPROT, INIT and MSEL fields are blocked. Reset type: SYSRSn

### 3.15.7.10 LSxCOMMIT Register (Offset = 22h) [Reset = 0000000h]

LSxCOMMIT is shown in [Figure 3-101](#) and described in [Table 3-115](#).

Return to the [Summary Table](#).

Local Shared RAM Config Lock Commit Register

**Figure 3-101. 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	COMMIT_LS1	COMMIT_LS0
R/WOnce-0h							

**Table 3-115. LSxCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R/WOnce	0h	Reserved
6	RESERVED	R/WOnce	0h	Reserved
5	RESERVED	R/WOnce	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	RESERVED	R/WOnce	0h	Reserved
2	RESERVED	R/WOnce	0h	Reserved
1	COMMIT_LS1	R/WOnce	0h	Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT, INIT and MSEL fields are permanently blocked. Reset type: SYSRSn
0	COMMIT_LS0	R/WOnce	0h	Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM: 0: Write to ACCPROT, INIT and MSEL fields are allowed based on value of lock field in LSxLOCK register. 1: Write to ACCPROT, INIT and MSEL fields are permanently blocked. Reset type: SYSRSn

### 3.15.7.11 LSxACCPROT0 Register (Offset = 28h) [Reset = 0000000h]

LSxACCPROT0 is shown in [Figure 3-102](#) and described in [Table 3-116](#).

Return to the [Summary Table](#).

Local Shared RAM Config Register 0

**Figure 3-102. LSxACCPROT0 Register**

31	30	29	28	27	26	25	24
RESERVED						RESERVED	RESERVED
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED						RESERVED	RESERVED
R-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED						CPUWRPROT_ LS1	FETCHPROT_ LS1
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						CPUWRPROT_ LS0	FETCHPROT_ LS0
R-0h						R/W-0h	R/W-0h

**Table 3-116. LSxACCPROT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-18	RESERVED	R	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15-10	RESERVED	R	0h	Reserved
9	CPUWRPROT_LS1	R/W	0h	CPU WR Protection For LS1 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
8	FETCHPROT_LS1	R/W	0h	Fetch Protection For LS1 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	CPUWRPROT_LS0	R/W	0h	CPU WR Protection For LS0 RAM: 0: CPU Writes are allowed. 1: CPU Writes are blocked. Reset type: SYSRSn
0	FETCHPROT_LS0	R/W	0h	Fetch Protection For LS0 RAM: 0: CPU Fetch are allowed. 1: CPU Fetch are blocked. Reset type: SYSRSn

### 3.15.7.12 LSxTEST Register (Offset = 30h) [Reset = 0000000h]

LSxTEST is shown in [Figure 3-103](#) and described in [Table 3-117](#).

Return to the [Summary Table](#).

Local Shared RAM TEST Register

**Figure 3-103. 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		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		TEST_LS1		TEST_LS0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 3-117. LSxTEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R/W	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R/W	0h	Reserved
3-2	TEST_LS1	R/W	0h	Selects the different 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: Same as functional mode, but interrupt/NMI is not generated on error. Note: Any non zero value would enable CPU writes over-riding write access protection if any and will not generate a access protection violation. Reset type: SYSRSn
1-0	TEST_LS0	R/W	0h	Selects the different modes for LS0 RAM: 00: Functional Mode. 01: Writes are allowed to data bits only. No write to parity bits. 10: Writes are allowed to parity bits only. No write to data bits. 11: Same as functional mode, but interrupt/NMI is not generated on error. Note: Any non zero value would enable CPU writes over-riding write access protection if any and will not generate a access protection violation. Reset type: SYSRSn

### 3.15.7.13 LSxINIT Register (Offset = 32h) [Reset = 0000000h]

LSxINIT is shown in [Figure 3-104](#) and described in [Table 3-118](#).

Return to the [Summary Table](#).

Local Shared RAM Init Register

**Figure 3-104. 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	INIT_LS1	INIT_LS0
R-0/W1S-0h							

**Table 3-118. LSxINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	INIT_LS1	R-0/W1S	0h	RAM Initialization control for LS1 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn
0	INIT_LS0	R-0/W1S	0h	RAM Initialization control for LS0 RAM: 0: None. 1: Start RAM Initialization. Reset type: SYSRSn

### 3.15.7.14 LSxINITDONE Register (Offset = 34h) [Reset = 0000000h]

LSxINITDONE is shown in [Figure 3-105](#) and described in [Table 3-119](#).

Return to the [Summary Table](#).

Local Shared RAM InitDone Status Register

**Figure 3-105. 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	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	INITDONE_LS1	INITDONE_LS0
R-0h	R-0h						

**Table 3-119. LSxINITDONE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	INITDONE_LS1	R	0h	RAM Initialization status for LS1 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn
0	INITDONE_LS0	R	0h	RAM Initialization status for LS0 RAM: 0: RAM Initialization is not done. 1: RAM Initialization is done. Reset type: SYSRSn

### 3.15.7.15 LSxRAMTEST\_LOCK Register (Offset = 36h) [Reset = 0000000h]

LSxRAMTEST\_LOCK is shown in [Figure 3-106](#) and described in [Table 3-120](#).

Return to the [Summary Table](#).

Lock register to LSx RAM TEST registers

**Figure 3-106. LSxRAMTEST\_LOCK 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								RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	RESE RVED	LS1	LS0	
R-0h								R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-120. LSxRAMTEST\_LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	A value of 0xa5a5 to this field is simultaneously required for the writes to the rest of the fields of this register to succeed, Reset type: SYSRSn
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	LS1	R/W	0h	0: Allows writes to LSxTEST.TEST_LS1 field. 1: Blocks writes to LSxTEST.TEST_LS1 field. Reset type: SYSRSn
0	LS0	R/W	0h	0: Allows writes to LSxTEST.TEST_LS0 field. 1: Blocks writes to LSxTEST.TEST_LS0 field. Reset type: SYSRSn

### 3.15.7.16 ROM\_LOCK Register (Offset = A0h) [Reset = 0000000h]

ROM\_LOCK is shown in [Figure 3-107](#) and described in [Table 3-121](#).

Return to the [Summary Table](#).

ROM Config Lock Register

**Figure 3-107. ROM\_LOCK Register**

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

**Table 3-121. ROM\_LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	A value of 0xa5a5 to this field is simultaneously required for the writes to the rest of the fields of this register to succeed, Reset type: SYSRSn
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	LOCK_BOOTROM	R/W	0h	Locks write access to test control fields (TEST and FORCE_ERROR) of BOOTROM 0: Write access allowed 1: Write access blocked Reset type: SYSRSn

### 3.15.7.17 ROM\_TEST Register (Offset = A2h) [Reset = 0000000h]

ROM\_TEST is shown in [Figure 3-108](#) and described in [Table 3-122](#).

Return to the [Summary Table](#).

ROM TEST Register

**Figure 3-108. ROM\_TEST Register**

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

**Table 3-122. ROM\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R/W	0h	Reserved
3-2	RESERVED	R/W	0h	Reserved
1-0	TEST_BOOTROM	R/W	0h	Selects the different modes for BOOTROM: 00: Functional Mode. 01: same as '00' but Parity check on data read is disabled (for debug) 10: Parity Bits are visible on memory map (for debug) 11: Same as '00' but NMI is not generated on errors, used for diagnostics. (for diagnostics) Reset type: SYSRSn

### 3.15.7.18 ROM\_FORCE\_ERROR Register (Offset = A4h) [Reset = 0000000h]

ROM\_FORCE\_ERROR is shown in [Figure 3-109](#) and described in [Table 3-123](#).

Return to the [Summary Table](#).

ROM Force Error register

**Figure 3-109. ROM\_FORCE\_ERROR Register**

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

**Table 3-123. ROM\_FORCE\_ERROR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	FORCE_BOOTROM_ERROR	R/W	0h	Force parity error by feeding inverted Parity bit to Parity checking logic. Reset type: SYSRSn

### 3.15.8 MEMORY\_ERROR\_REGS Registers

Table 3-124 lists the memory-mapped registers for the MEMORY\_ERROR\_REGS registers. All register offset addresses not listed in Table 3-124 should be considered as reserved locations and the register contents should not be modified.

**Table 3-124. MEMORY\_ERROR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	UCERRFLG	Uncorrectable Error Flag Register		<a href="#">Go</a>
2h	UCERRSET	Uncorrectable Error Flag Set Register	EALLOW	<a href="#">Go</a>
4h	UCERRCLR	Uncorrectable Error Flag Clear Register	EALLOW	<a href="#">Go</a>
6h	UCCPUREADDR	Uncorrectable CPU Read Error Address		<a href="#">Go</a>
1Ch	FLUCERRSTATUS	Flash read uncorrectable ecc err status		<a href="#">Go</a>
1Eh	FLCERRSTATUS	Flash read correctable ecc err status		<a href="#">Go</a>
20h	CERRFLG	Correctable Error Flag Register		<a href="#">Go</a>
22h	CERRSET	Correctable Error Flag Set Register	EALLOW	<a href="#">Go</a>
24h	CERRCLR	Correctable Error Flag Clear Register	EALLOW	<a href="#">Go</a>
26h	CCPUREADDR	Correctable CPU Read Error Address		<a href="#">Go</a>
2Eh	CERRCNT	Correctable Error Count Register		<a href="#">Go</a>
30h	CERRTHRES	Correctable Error Threshold Value Register	EALLOW	<a href="#">Go</a>
32h	CEINTFLG	Correctable Error Interrupt Flag Status Register		<a href="#">Go</a>
34h	CEINTCLR	Correctable Error Interrupt Flag Clear Register	EALLOW	<a href="#">Go</a>
36h	CEINTSET	Correctable Error Interrupt Flag Set Register	EALLOW	<a href="#">Go</a>
38h	CEINTEN	Correctable Error Interrupt Enable Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-125 shows the codes that are used for access types in this section.

**Table 3-125. MEMORY\_ERROR\_REGS Access Type Codes**

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

### 3.15.8.1 UCERRFLG Register (Offset = 0h) [Reset = 0000000h]

UCERRFLG is shown in [Figure 3-110](#) and described in [Table 3-126](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Register

**Figure 3-110. 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	RESERVED	RESERVED	RESERVED	RESERVED	CPURDERR
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-126. UCERRFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	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	CPURDERR	R	0h	CPU Uncorrectable Read Error Flag 0: No Error. 1: Uncorrectable error occurred during CPU read. Reset type: SYSRSn

### 3.15.8.2 UCERRSET Register (Offset = 2h) [Reset = 0000000h]

UCERRSET is shown in [Figure 3-111](#) and described in [Table 3-127](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Set Register

**Figure 3-111. 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	RESERVED	RESERVED	RESERVED	RESERVED	CPURDERR
R-0h		R-0/W1S-0h	R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-127. UCERRSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	RESERVED	R-0/W1S	0h	Reserved
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn

### 3.15.8.3 UCERRCLR Register (Offset = 4h) [Reset = 0000000h]

UCERRCLR is shown in [Figure 3-112](#) and described in [Table 3-128](#).

Return to the [Summary Table](#).

Uncorrectable Error Flag Clear Register

**Figure 3-112. 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	RESERVED	RESERVED	RESERVED	RESERVED	CPURDERR
R-0h		R-0/W1S-0h	R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-128. UCERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-6	RESERVED	R	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	RESERVED	R-0/W1S	0h	Reserved
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in UCERRFLG register will be cleared. Reset type: SYSRSn

### 3.15.8.4 UCCPUREADDR Register (Offset = 6h) [Reset = 0000000h]

UCCPUREADDR is shown in [Figure 3-113](#) and described in [Table 3-129](#).

Return to the [Summary Table](#).

Uncorrectable CPU Read Error Address

**Figure 3-113. UCCPUREADDR Register**

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

**Table 3-129. UCCPUREADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UCCPUREADDR	R	0h	This register captures the address location for which CPU read/fetch access resulted in uncorrectable ECC/Parity error. Reset type: SYSRSn

### 3.15.8.5 FLUCERRSTATUS Register (Offset = 1Ch) [Reset = 0000000h]

FLUCERRSTATUS is shown in [Figure 3-114](#) and described in [Table 3-130](#).

Return to the [Summary Table](#).

Flash read uncorrectable ecc err status

**Figure 3-114. FLUCERRSTATUS 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						DIAG_H_FAIL	UNC_ERR_H
R-0h						R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED						DIAG_L_FAIL	UNC_ERR_L
R-0h						R-0h	R-0h

**Table 3-130. FLUCERRSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	DIAG_H_FAIL	R	0h	Status of redundant and functional ECC logic comparison check. 0 : Status of redundant and functional ECC logic comparison passed. 1 : Status of redundant and functional ECC logic comparison failed. Note : * The field gets updated along with UNC_ERR_H * This flag is cleared by writing a 1 to UCERRCLR.CPURDERR flag * UCERRSET.CPURDERR has no effect on this flag. Reset type: SYSRSn
8	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. Note : * This flag is cleared by writing a 1 to UCERRCLR.CPURDERR flag * UCERRSET.CPURDERR has no effect on this flag. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	DIAG_L_FAIL	R	0h	Status of redundant and functional ECC logic comparison check. 0 : Status of redundant and functional ECC logic comparison passed. 1 : Status of redundant and functional ECC logic comparison failed. Note : * The field gets updated along with UNC_ERR_L * This flag is cleared by writing a 1 to UCERRCLR.CPURDERR flag * UCERRSET.CPURDERR has no effect on this flag. Reset type: SYSRSn
0	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. Note : * This flag is cleared by writing a 1 to UCERRCLR.CPURDERR flag * UCERRSET.CPURDERR has no effect on this flag. Reset type: SYSRSn

### 3.15.8.6 FLCERRSTATUS Register (Offset = 1Eh) [Reset = 0000000h]

FLCERRSTATUS is shown in [Figure 3-115](#) and described in [Table 3-131](#).

Return to the [Summary Table](#).

Flash read correctable ecc err status

**Figure 3-115. FLCERRSTATUS Register**

31	30	29	28	27	26	25	24
RESERVED		ERR_TYPE_H	ERR_POS_H				
R-0h		R-0h	R-0h				
23	22	21	20	19	18	17	16
ERR_POS_H	ERR_TYPE_L	ERR_POS_L					
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	FAIL_1_H	FAIL_0_H	RESERVED	FAIL_1_L	FAIL_0_L
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-131. FLCERRSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29	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. Note : * This field is cleared by writing a 1 to CERRCLR.CPURDERR flag * CERRSET.CPURDERR has no effect on this field. Reset type: SYSRSn
28-23	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. Note : * This field is cleared by writing a 1 to CERRCLR.CPURDERR flag * CERRSET.CPURDERR has no effect on this field. Reset type: SYSRSn
22	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. Note : * This field is cleared by writing a 1 to CERRCLR.CPURDERR flag * CERRSET.CPURDERR has no effect on this field. Reset type: SYSRSn

**Table 3-131. FLCERRSTATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
21-16	ERR_POS_L	R	0h	<p>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.</p> <p>Note :</p> <ul style="list-style-type: none"> <li>* This field is cleared by writing a 1 to CERRCLR.CPURDERR flag</li> <li>* CERRSET.CPURDERR has no effect on this field.</li> </ul> <p>Reset type: SYSRSn</p>
15-6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	FAIL_1_H	R	0h	<p>Fail on 1.</p> <p>0 Fail on 1 single bit error did not occur in upper 64bits of a 128-bit aligned address.</p> <p>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.</p> <p>Note :</p> <ul style="list-style-type: none"> <li>* This flag is cleared by writing a 1 to CERRCLR.CPURDERR flag</li> <li>* CERRSET.CPURDERR has no effect on this flag.</li> </ul> <p>Reset type: SYSRSn</p>
3	FAIL_0_H	R	0h	<p>Fail on 0.</p> <p>0 Fail on 0 single bit error did not occur in upper 64bits of a 128-bit aligned address.</p> <p>1 A value of 1 would indicate that a single bit error occurred in upper 64bits of a 128-bit aligned address and the corrected value was 0.</p> <p>Note :</p> <ul style="list-style-type: none"> <li>* This flag is cleared by writing a 1 to CERRCLR.CPURDERR flag</li> <li>* CERRSET.CPURDERR has no effect on this flag.</li> </ul> <p>Reset type: SYSRSn</p>
2	RESERVED	R	0h	Reserved
1	FAIL_1_L	R	0h	<p>Fail on 1.</p> <p>0 Fail on 1 single bit error did not occur in lower 64bits of a 128-bit aligned address.</p> <p>1 A value of 1 would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 1.</p> <p>Note :</p> <ul style="list-style-type: none"> <li>* This flag is cleared by writing a 1 to CERRCLR.CPURDERR flag</li> <li>* CERRSET.CPURDERR has no effect on this flag.</li> </ul> <p>Reset type: SYSRSn</p>
0	FAIL_0_L	R	0h	<p>Fail on 0.</p> <p>0 Fail on 0 single bit error did not occur in lower 64bits of 128-bit data.</p> <p>1 Would indicate that a single bit error occurred in lower 64bits of a 128-bit aligned address and the corrected value was 0.</p> <p>Note :</p> <ul style="list-style-type: none"> <li>* This flag is cleared by writing a 1 to CERRCLR.CPURDERR flag</li> <li>* CERRSET.CPURDERR has no effect on this flag.</li> </ul> <p>Reset type: SYSRSn</p>

### 3.15.8.7 CERRFLG Register (Offset = 20h) [Reset = 0000000h]

CERRFLG is shown in [Figure 3-116](#) and described in [Table 3-132](#).

Return to the [Summary Table](#).

Correctable Error Flag Register

**Figure 3-116. 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	RESERVED	RESERVED	CPURDERR
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 3-132. 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	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	CPURDERR	R	0h	CPU Correctable Read Error Flag 0: No Error. 1: Correctable error occurred during CPU read. Reset type: SYSRSn

### 3.15.8.8 CERRSET Register (Offset = 22h) [Reset = 0000000h]

CERRSET is shown in [Figure 3-117](#) and described in [Table 3-133](#).

Return to the [Summary Table](#).

Correctable Error Flag Set Register

**Figure 3-117. 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	RESERVED	RESERVED	CPURDERR
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-133. CERRSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	RESERVED	R-0/W1S	0h	Reserved
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be set and interrupt will be generated if enabled.. Reset type: SYSRSn

### 3.15.8.9 CERRCLR Register (Offset = 24h) [Reset = 0000000h]

CERRCLR is shown in [Figure 3-118](#) and described in [Table 3-134](#).

Return to the [Summary Table](#).

Correctable Error Flag Clear Register

**Figure 3-118. 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	RESERVED	RESERVED	CPURDERR
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-134. CERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	RESERVED	R-0/W1S	0h	Reserved
1	RESERVED	R-0/W1S	0h	Reserved
0	CPURDERR	R-0/W1S	0h	0: No action. 1: CPU Read Error Flag in CERRFLG register will be cleared. Reset type: SYSRSn

### 3.15.8.10 CCPUREADDR Register (Offset = 26h) [Reset = 0000000h]

CCPUREADDR is shown in [Figure 3-119](#) and described in [Table 3-135](#).

Return to the [Summary Table](#).

Correctable CPU Read Error Address

**Figure 3-119. CCPUREADDR Register**

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

**Table 3-135. CCPUREADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CCPUREADDR	R	0h	This register captures the address location for which CPU read/fetch access resulted in correctable ECC error. Reset type: SYSRSn

### 3.15.8.11 CERRCNT Register (Offset = 2Eh) [Reset = 0000000h]

CERRCNT is shown in [Figure 3-120](#) and described in [Table 3-136](#).

Return to the [Summary Table](#).

Correctable Error Count Register

**Figure 3-120. CERRCNT Register**

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

**Table 3-136. CERRCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CERRCNT	R	0h	This register holds the count of how many times correctable error occurred. Reset type: SYSRSn

### 3.15.8.12 CERRTHRES Register (Offset = 30h) [Reset = 0000000h]

CERRTHRES is shown in [Figure 3-121](#) and described in [Table 3-137](#).

Return to the [Summary Table](#).

Correctable Error Threshold Value Register

**Figure 3-121. 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
RESERVED																CERRTHRES															
R-0h																R/W-0h															

**Table 3-137. CERRTHRES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CERRTHRES	R/W	0h	When value in CERRCNT register is greater than value configured in this register, correctable interrupt gets generated, if enabled. Reset type: SYSRSn

### 3.15.8.13 CEINTFLG Register (Offset = 32h) [Reset = 0000000h]

CEINTFLG is shown in [Figure 3-122](#) and described in [Table 3-138](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Status Register

**Figure 3-122. CEINTFLG Register**

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

**Table 3-138. CEINTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTFLAG	R	0h	Total corrected error count exceeded threshold Flag 0: Total correctable errors < Threshold value configured in CERRTHRES register. 1: Total correctable errors >= Threshold value configured in CERRTHRES register. Reset type: SYSRSn

### 3.15.8.14 CEINTCLR Register (Offset = 34h) [Reset = 0000000h]

CEINTCLR is shown in [Figure 3-123](#) and described in [Table 3-139](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Clear Register

**Figure 3-123. CEINTCLR Register**

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

**Table 3-139. CEINTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTCLR	R-0/W1S	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be cleared. Reset type: SYSRSn

### 3.15.8.15 CEINTSET Register (Offset = 36h) [Reset = 0000000h]

CEINTSET is shown in [Figure 3-124](#) and described in [Table 3-140](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Flag Set Register

**Figure 3-124. CEINTSET Register**

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

**Table 3-140. CEINTSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTSET	R-0/W1S	0h	0: No action. 1: Total corrected error count exceeded flag in CEINTFLG register will be set and interrupt will be generated if enabled. Reset type: SYSRSn

### 3.15.8.16 CEINTEN Register (Offset = 38h) [Reset = 0000000h]

CEINTEN is shown in [Figure 3-125](#) and described in [Table 3-141](#).

Return to the [Summary Table](#).

Correctable Error Interrupt Enable Register

**Figure 3-125. CEINTEN Register**

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

**Table 3-141. CEINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-1	RESERVED	R	0h	Reserved
0	CEINTEN	R/W	0h	0: Correctable Error Interrupt is disabled. 1: Correctable Error Interrupt is enabled. Reset type: SYSRSn

### 3.15.9 NMI\_INTRUPT\_REGS Registers

Table 3-142 lists the memory-mapped registers for the NMI\_INTRUPT\_REGS registers. All register offset addresses not listed in Table 3-142 should be considered as reserved locations and the register contents should not be modified.

**Table 3-142. NMI\_INTRUPT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	NMICFG	NMI Configuration Register	EALLOW	<a href="#">Go</a>
1h	NMIFLG	NMI Flag Register (SYSRsn Clear)		<a href="#">Go</a>
2h	NMIFLGCLR	NMI Flag Clear Register	EALLOW	<a href="#">Go</a>
3h	NMIFLGFRC	NMI Flag Force Register	EALLOW	<a href="#">Go</a>
4h	NMIWDCNT	NMI Watchdog Counter Register		<a href="#">Go</a>
5h	NMIWDPD	NMI Watchdog Period Register	EALLOW	<a href="#">Go</a>
6h	NMISHDFLG	NMI Shadow Flag Register		<a href="#">Go</a>
7h	ERRORSTS	Error pin status		<a href="#">Go</a>
8h	ERRORSTSCLR	ERRORSTS clear register	EALLOW	<a href="#">Go</a>
9h	ERRORSTSFRC	ERRORSTS force register	EALLOW	<a href="#">Go</a>
Ah	ERRORCTL	Error pin control register	EALLOW	<a href="#">Go</a>
Bh	ERRORLOCK	Lock register to Error pin registers.	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-143 shows the codes that are used for access types in this section.

**Table 3-143. NMI\_INTRUPT\_REGS Access Type Codes**

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

### 3.15.9.1 NMICFG Register (Offset = 0h) [Reset = 0000h]

NMICFG is shown in [Figure 3-126](#) and described in [Table 3-144](#).

Return to the [Summary Table](#).

NMI Configuration Register

**Figure 3-126. NMICFG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							NMIE
R-0-0h							R/W1S-0h

**Table 3-144. NMICFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	NMIE	R/W1S	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 Reset type: SYSRSn

### 3.15.9.2 NMIFLG Register (Offset = 1h) [Reset = 0000h]

NMIFLG is shown in [Figure 3-127](#) and described in [Table 3-145](#).

Return to the [Summary Table](#).

NMI Flag Register (SYSRSn Clear)

**Figure 3-127. NMIFLG Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	SWERR	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	UNCERR	CLOCKFAIL	NMIINT
R-0h	R-0h						

**Table 3-145. NMIFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R-0	0h	Reserved
14	RESERVED	R	0h	Reserved
13	SWERR	R	0h	SW Error Force NMI Flag: This bit indicates if an NMI was forced through the NMIFLGFRC register. This bit can only be cleared by the user writing to the respective bit in the NMIFLGCLR register or by an SYSRSn reset: 0 No SW Error force Generated 1 SW Error NMI is forced by SW. No further NMI pulses are generated until this flag is cleared by the user. Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	UNCERR	R	0h	Flash/RAM/ROM Uncorrectable Error NMI Flag: This bit indicates if an uncorrectable error occurred on a memory 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 SYSRSn reset: 0, No uncorrectable error condition pending 1, uncorrectable error condition generated Reset type: SYSRSn
1	CLOCKFAIL	R	0h	Clock Fail Interrupt Flag: These bits indicates 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 SYSRSn reset: 0, No CLOCKFAIL Condition Pending 1, CLOCKFAIL Condition Generated Reset type: SYSRSn

**Table 3-145. NMIFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 SYSRSn 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. Reset type: SYSRSn

### 3.15.9.3 NMIFLGCLR Register (Offset = 2h) [Reset = 0000h]

NMIFLGCLR is shown in [Figure 3-128](#) and described in [Table 3-146](#).

Return to the [Summary Table](#).

NMI Flag Clear Register

**Figure 3-128. NMIFLGCLR Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	SWERR	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h	R-0/W1S-0h						
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	UNCERR	CLOCKFAIL	NMIINT
R-0/W1S-0h							

**Table 3-146. NMIFLGCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R-0	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	SWERR	R-0/W1S	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. Reset type: SYSRSn
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	UNCERR	R-0/W1S	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. Reset type: SYSRSn
1	CLOCKFAIL	R-0/W1S	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. Reset type: SYSRSn

**Table 3-146. NMIFLGCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	NMIINT	R-0/W1S	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. Reset type: SYSRSn

### 3.15.9.4 NMIFLGFR Register (Offset = 3h) [Reset = 0000h]

NMIFLGFR is shown in [Figure 3-129](#) and described in [Table 3-147](#).

Return to the [Summary Table](#).

NMI Flag Force Register

**Figure 3-129. NMIFLGFR Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	SWERR	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h	R-0/W1S-0h						
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	UNCERR	CLOCKFAIL	RESERVED
R-0/W1S-0h	R-0-0h						

**Table 3-147. NMIFLGFR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R-0	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	SWERR	R-0/W1S	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. Reset type: SYSRSn
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	UNCERR	R-0/W1S	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. Reset type: SYSRSn
1	CLOCKFAIL	R-0/W1S	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. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 3.15.9.5 NMIWDCNT Register (Offset = 4h) [Reset = 0000h]

NMIWDCNT is shown in [Figure 3-130](#) and described in [Table 3-148](#).

Return to the [Summary Table](#).

NMI Watchdog Counter Register

**Figure 3-130. NMIWDCNT Register**

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

**Table 3-148. 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> <p>Reset type: SYSRStn</p>

### 3.15.9.6 NMIWDPRD Register (Offset = 5h) [Reset = FFFFh]

NMIWDPRD is shown in [Figure 3-131](#) and described in [Table 3-149](#).

Return to the [Summary Table](#).

NMI Watchdog Period Register

**Figure 3-131. NMIWDPRD Register**

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

**Table 3-149. 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 than the current counter value will automatically force an NMIRSn and hence reset the watchdog counter.</p> <p>Reset type: SYSRSn</p>

### 3.15.9.7 NMISHDFLG Register (Offset = 6h) [Reset = 0000h]

NMISHDFLG is shown in [Figure 3-132](#) and described in [Table 3-150](#).

Return to the [Summary Table](#).

NMI Shadow Flag Register

**Figure 3-132. NMISHDFLG Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	SWERR	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	UNCERR	CLOCKFAIL	RESERVED
R-0h	R-0-0h						

**Table 3-150. NMISHDFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R-0	0h	Reserved
14	RESERVED	R	0h	Reserved
13	SWERR	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 reset only by PORESETn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean. Reset type: PORESETn
12	RESERVED	R	0h	Reserved
11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	UNCERR	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 reset only by PORESETn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean. Reset type: PORESETn
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 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 reset only by PORESETn. Notes: [1] This register is added to keep the definition of System Control Reset Cause Register Clean. Reset type: PORESETn

**Table 3-150. NMISHDFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RESERVED	R-0	0h	Reserved

### 3.15.9.8 ERRORSTS Register (Offset = 7h) [Reset = 0000h]

ERRORSTS is shown in [Figure 3-133](#) and described in [Table 3-151](#).

Return to the [Summary Table](#).

Error pin status

**Figure 3-133. ERRORSTS Register**

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

**Table 3-151. ERRORSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R-0	0h	Reserved
1	PINSTS	R	0h	0, Error Pin is 0 1, Error Pin is 1 Reset type: PORESETn
0	ERROR	R	0h	0, None of the error sources were triggered. 1, One or more of the error sources triggered, or ERRORSTS.ERROR was set by a write of 1 to ERRORSTSFRC.ERROR bit. Once set, the ERROR flag can be cleared by writing 1 to ERRORSTSCLR.ERROR bit. Following are the events/triggers which can set this bit: 1. nmi interrupt on C28x 2. Watchdog reset 3. Error on a Pie vector fetch 4. Efuse error On a read of this bit, the pin Error pin state will be returned. Reset type: PORESETn

### 3.15.9.9 ERRORSTSCLR Register (Offset = 8h) [Reset = 0000h]

ERRORSTSCLR is shown in [Figure 3-134](#) and described in [Table 3-152](#).

Return to the [Summary Table](#).

ERRORSTS clear register

**Figure 3-134. ERRORSTSCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							ERROR
R-0-0h							R-0/W1S-0h

**Table 3-152. ERRORSTSCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	ERROR	R-0/W1S	0h	0, No effect 1, ERRORSTS.ERROR is cleared to 0 Reset type: PORESETn

### 3.15.9.10 ERRORSTSFRC Register (Offset = 9h) [Reset = 0000h]

ERRORSTSFRC is shown in [Figure 3-135](#) and described in [Table 3-153](#).

Return to the [Summary Table](#).

ERRORSTS force register

**Figure 3-135. ERRORSTSFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							ERROR
R-0-0h							R-0/W1S-0h

**Table 3-153. ERRORSTSFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	ERROR	R-0/W1S	0h	0, No effect 1, ERRORSTS.ERROR is set to 1 Reset type: PORESETn

### 3.15.9.11 ERRORCTL Register (Offset = Ah) [Reset = 0000h]

ERRORCTL is shown in [Figure 3-136](#) and described in [Table 3-154](#).

Return to the [Summary Table](#).

Error pin control register

**Figure 3-136. ERRORCTL Register**

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

**Table 3-154. ERRORCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	ERRORPOLSEL	R/W	0h	0, If ERRORSTS.ERROR is 1, Error pin will be driven with a value of 0, else 1. 1, If ERRORSTS.ERROR is 1, Error pin will be driven with a value of 1, else 0. Reset type: PORESETn

### 3.15.9.12 ERRORLOCK Register (Offset = Bh) [Reset = 0000h]

ERRORLOCK is shown in [Figure 3-137](#) and described in [Table 3-155](#).

Return to the [Summary Table](#).

Lock register to Error pin registers.

**Figure 3-137. ERRORLOCK Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED							ERRORCTL
R-0-0h							R/WOnce-0h

**Table 3-155. ERRORLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	ERRORCTL	R/WOnce	0h	0, Writes to ERRORCTL register allowed. 1, Writes to ERRORCTL register is blocked. Writes of 0 to this bit has no effect. Write of 1 will set this bit, cleared only on a SYSRSn. Reset type: SYSRSn

### 3.15.10 PIE\_CTRL\_REGS Registers

Table 3-156 lists the memory-mapped registers for the PIE\_CTRL\_REGS registers. All register offset addresses not listed in Table 3-156 should be considered as reserved locations and the register contents should not be modified.

**Table 3-156. PIE\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	PIECTRL	ePIE Control Register		<a href="#">Go</a>
1h	PIEACK	Interrupt Acknowledge Register		<a href="#">Go</a>
2h	PIEIER1	Interrupt Group 1 Enable Register		<a href="#">Go</a>
3h	PIEIFR1	Interrupt Group 1 Flag Register		<a href="#">Go</a>
4h	PIEIER2	Interrupt Group 2 Enable Register		<a href="#">Go</a>
5h	PIEIFR2	Interrupt Group 2 Flag Register		<a href="#">Go</a>
6h	PIEIER3	Interrupt Group 3 Enable Register		<a href="#">Go</a>
7h	PIEIFR3	Interrupt Group 3 Flag Register		<a href="#">Go</a>
8h	PIEIER4	Interrupt Group 4 Enable Register		<a href="#">Go</a>
9h	PIEIFR4	Interrupt Group 4 Flag Register		<a href="#">Go</a>
Ah	PIEIER5	Interrupt Group 5 Enable Register		<a href="#">Go</a>
Bh	PIEIFR5	Interrupt Group 5 Flag Register		<a href="#">Go</a>
Ch	PIEIER6	Interrupt Group 6 Enable Register		<a href="#">Go</a>
Dh	PIEIFR6	Interrupt Group 6 Flag Register		<a href="#">Go</a>
Eh	PIEIER7	Interrupt Group 7 Enable Register		<a href="#">Go</a>
Fh	PIEIFR7	Interrupt Group 7 Flag Register		<a href="#">Go</a>
10h	PIEIER8	Interrupt Group 8 Enable Register		<a href="#">Go</a>
11h	PIEIFR8	Interrupt Group 8 Flag Register		<a href="#">Go</a>
12h	PIEIER9	Interrupt Group 9 Enable Register		<a href="#">Go</a>
13h	PIEIFR9	Interrupt Group 9 Flag Register		<a href="#">Go</a>
14h	PIEIER10	Interrupt Group 10 Enable Register		<a href="#">Go</a>
15h	PIEIFR10	Interrupt Group 10 Flag Register		<a href="#">Go</a>
16h	PIEIER11	Interrupt Group 11 Enable Register		<a href="#">Go</a>
17h	PIEIFR11	Interrupt Group 11 Flag Register		<a href="#">Go</a>
18h	PIEIER12	Interrupt Group 12 Enable Register		<a href="#">Go</a>
19h	PIEIFR12	Interrupt Group 12 Flag Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-157 shows the codes that are used for access types in this section.

**Table 3-157. PIE\_CTRL\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1S	W 1S	Write 1 to set
Reset or Default Value		

**Table 3-157. PIE\_CTRL\_REGS Access Type Codes  
(continued)**

Access Type	Code	Description
-n		Value after reset or the default value

### 3.15.10.1 PIECTRL Register (Offset = 0h) [Reset = 0000h]

PIECTRL is shown in [Figure 3-138](#) and described in [Table 3-158](#).

Return to the [Summary Table](#).

ePIE Control Register

**Figure 3-138. PIECTRL Register**

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

**Table 3-158. PIECTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	PIEVECT	R	0h	These bits indicate the vector address of the vector fetched from the ePIE vector table. The least significant bit of the address is ignored and only bits 1 to 15 of the address are shown. The vector value can be read by the user to determine which interrupt generated the vector fetch. Note: When a NMI is serviced, the PIEVECT bit-field does not reflect the vector as it does for other interrupts. Reset type: SYSRSn
0	ENPIE	R/W	0h	Enable vector fetching from ePIE block. This bit must be set to 1 for peripheral interrupts to work. All ePIE registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the ePIE block is disabled. Reset type: SYSRSn

### 3.15.10.2 PIEACK Register (Offset = 1h) [Reset = 0000h]

PIEACK is shown in [Figure 3-139](#) and described in [Table 3-159](#).

Return to the [Summary Table](#).

#### Acknowledge Register

When an interrupt propagates from the ePIE to a CPU interrupt line, the interrupt group's PIEACK bit is set. This prevents other interrupts in that group from propagating to the CPU while the first interrupt is handled. Writing a 1 to a PIEACK bit clears it and allows another interrupt from the corresponding group to propagate. ISRs for PIE interrupts should clear the group's PIEACK bit before returning from the interrupt.

Writes of 0 are ignored.

**Figure 3-139. PIEACK Register**

15	14	13	12	11	10	9	8
RESERVED				ACK12	ACK11	ACK10	ACK9
R-0-0h				R/W1S-0h	R/W1S-0h	R/W1S-0h	R/W1S-0h
7	6	5	4	3	2	1	0
ACK8	ACK7	ACK6	ACK5	ACK4	ACK3	ACK2	ACK1
R/W1S-0h							

**Table 3-159. PIEACK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11	ACK12	R/W1S	0h	Acknowledge PIE Interrupt Group 12 Reset type: SYSRSn
10	ACK11	R/W1S	0h	Acknowledge PIE Interrupt Group 11 Reset type: SYSRSn
9	ACK10	R/W1S	0h	Acknowledge PIE Interrupt Group 10 Reset type: SYSRSn
8	ACK9	R/W1S	0h	Acknowledge PIE Interrupt Group 9 Reset type: SYSRSn
7	ACK8	R/W1S	0h	Acknowledge PIE Interrupt Group 8 Reset type: SYSRSn
6	ACK7	R/W1S	0h	Acknowledge PIE Interrupt Group 7 Reset type: SYSRSn
5	ACK6	R/W1S	0h	Acknowledge PIE Interrupt Group 6 Reset type: SYSRSn
4	ACK5	R/W1S	0h	Acknowledge PIE Interrupt Group 5 Reset type: SYSRSn
3	ACK4	R/W1S	0h	Acknowledge PIE Interrupt Group 4 Reset type: SYSRSn
2	ACK3	R/W1S	0h	Acknowledge PIE Interrupt Group 3 Reset type: SYSRSn
1	ACK2	R/W1S	0h	Acknowledge PIE Interrupt Group 2 Reset type: SYSRSn
0	ACK1	R/W1S	0h	Acknowledge PIE Interrupt Group 1 Reset type: SYSRSn

### 3.15.10.3 PIEIER1 Register (Offset = 2h) [Reset = 0000h]

PIEIER1 is shown in [Figure 3-140](#) and described in [Table 3-160](#).

Return to the [Summary Table](#).

#### Interrupt Group 1 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-140. PIEIER1 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-160. PIEIER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 1.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 1.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 1.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 1.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 1.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 1.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 1.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 1.1 Reset type: SYSRSn

### 3.15.10.4 PIEIFR1 Register (Offset = 3h) [Reset = 0000h]

PIEIFR1 is shown in [Figure 3-141](#) and described in [Table 3-161](#).

Return to the [Summary Table](#).

#### Interrupt Group 1 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-141. PIEIFR1 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-161. PIEIFR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 1.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 1.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 1.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 1.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 1.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 1.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 1.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 1.1 Reset type: SYSRSn

### 3.15.10.5 PIEIER2 Register (Offset = 4h) [Reset = 0000h]

PIEIER2 is shown in [Figure 3-142](#) and described in [Table 3-162](#).

Return to the [Summary Table](#).

#### Interrupt Group 2 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-142. PIEIER2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-162. PIEIER2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 2.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 2.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 2.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 2.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 2.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 2.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 2.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 2.1 Reset type: SYSRSn

### 3.15.10.6 PIEIFR2 Register (Offset = 5h) [Reset = 0000h]

PIEIFR2 is shown in [Figure 3-143](#) and described in [Table 3-163](#).

Return to the [Summary Table](#).

#### Interrupt Group 2 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-143. PIEIFR2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-163. PIEIFR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 2.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 2.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 2.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 2.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 2.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 2.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 2.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 2.1 Reset type: SYSRSn

### 3.15.10.7 PIEIER3 Register (Offset = 6h) [Reset = 0000h]

PIEIER3 is shown in [Figure 3-144](#) and described in [Table 3-164](#).

Return to the [Summary Table](#).

#### Interrupt Group 3 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-144. PIEIER3 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-164. PIEIER3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 3.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 3.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 3.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 3.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 3.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 3.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 3.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 3.1 Reset type: SYSRSn

### 3.15.10.8 PIEIFR3 Register (Offset = 7h) [Reset = 0000h]

PIEIFR3 is shown in [Figure 3-145](#) and described in [Table 3-165](#).

Return to the [Summary Table](#).

#### Interrupt Group 3 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-145. PIEIFR3 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-165. PIEIFR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 3.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 3.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 3.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 3.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 3.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 3.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 3.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 3.1 Reset type: SYSRSn

### 3.15.10.9 PIEIER4 Register (Offset = 8h) [Reset = 0000h]

PIEIER4 is shown in [Figure 3-146](#) and described in [Table 3-166](#).

Return to the [Summary Table](#).

#### Interrupt Group 4 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-146. PIEIER4 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-166. PIEIER4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 4.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 4.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 4.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 4.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 4.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 4.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 4.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 4.1 Reset type: SYSRSn

### 3.15.10.10 PIEIFR4 Register (Offset = 9h) [Reset = 0000h]

PIEIFR4 is shown in [Figure 3-147](#) and described in [Table 3-167](#).

Return to the [Summary Table](#).

#### Interrupt Group 4 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-147. PIEIFR4 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-167. PIEIFR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 4.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 4.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 4.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 4.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 4.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 4.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 4.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 4.1 Reset type: SYSRSn

### 3.15.10.11 PIEIER5 Register (Offset = Ah) [Reset = 0000h]

PIEIER5 is shown in [Figure 3-148](#) and described in [Table 3-168](#).

Return to the [Summary Table](#).

#### Interrupt Group 5 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-148. PIEIER5 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-168. PIEIER5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 5.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 5.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 5.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 5.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 5.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 5.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 5.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 5.1 Reset type: SYSRSn

### 3.15.10.12 PIEIFR5 Register (Offset = Bh) [Reset = 0000h]

PIEIFR5 is shown in [Figure 3-149](#) and described in [Table 3-169](#).

Return to the [Summary Table](#).

#### Interrupt Group 5 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-149. PIEIFR5 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-169. PIEIFR5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 5.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 5.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 5.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 5.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 5.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 5.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 5.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 5.1 Reset type: SYSRSn

### 3.15.10.13 PIEIER6 Register (Offset = Ch) [Reset = 0000h]

PIEIER6 is shown in [Figure 3-150](#) and described in [Table 3-170](#).

Return to the [Summary Table](#).

#### Interrupt Group 6 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-150. PIEIER6 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-170. PIEIER6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 6.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 6.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 6.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 6.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 6.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 6.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 6.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 6.1 Reset type: SYSRSn

### 3.15.10.14 PIEIFR6 Register (Offset = Dh) [Reset = 0000h]

PIEIFR6 is shown in [Figure 3-151](#) and described in [Table 3-171](#).

Return to the [Summary Table](#).

#### Interrupt Group 6 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-151. PIEIFR6 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-171. PIEIFR6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 6.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 6.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 6.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 6.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 6.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 6.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 6.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 6.1 Reset type: SYSRSn

### 3.15.10.15 PIEIER7 Register (Offset = Eh) [Reset = 0000h]

PIEIER7 is shown in [Figure 3-152](#) and described in [Table 3-172](#).

Return to the [Summary Table](#).

#### Interrupt Group 7 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-152. PIEIER7 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-172. PIEIER7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 7.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 7.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 7.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 7.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 7.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 7.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 7.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 7.1 Reset type: SYSRSn

### 3.15.10.16 PIEIFR7 Register (Offset = Fh) [Reset = 0000h]

PIEIFR7 is shown in [Figure 3-153](#) and described in [Table 3-173](#).

Return to the [Summary Table](#).

#### Interrupt Group 7 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-153. PIEIFR7 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-173. PIEIFR7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 7.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 7.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 7.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 7.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 7.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 7.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 7.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 7.1 Reset type: SYSRSn

### 3.15.10.17 PIEIER8 Register (Offset = 10h) [Reset = 0000h]

PIEIER8 is shown in [Figure 3-154](#) and described in [Table 3-174](#).

Return to the [Summary Table](#).

#### Interrupt Group 8 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-154. PIEIER8 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-174. PIEIER8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 8.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 8.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 8.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 8.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 8.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 8.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 8.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 8.1 Reset type: SYSRSn

### 3.15.10.18 PIEIFR8 Register (Offset = 11h) [Reset = 0000h]

PIEIFR8 is shown in [Figure 3-155](#) and described in [Table 3-175](#).

Return to the [Summary Table](#).

#### Interrupt Group 8 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-155. PIEIFR8 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-175. PIEIFR8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 8.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 8.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 8.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 8.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 8.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 8.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 8.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 8.1 Reset type: SYSRSn

### 3.15.10.19 PIEIER9 Register (Offset = 12h) [Reset = 0000h]

PIEIER9 is shown in [Figure 3-156](#) and described in [Table 3-176](#).

Return to the [Summary Table](#).

#### Interrupt Group 9 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-156. PIEIER9 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-176. PIEIER9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 9.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 9.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 9.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 9.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 9.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 9.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 9.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 9.1 Reset type: SYSRSn

### 3.15.10.20 PIEIFR9 Register (Offset = 13h) [Reset = 0000h]

PIEIFR9 is shown in [Figure 3-157](#) and described in [Table 3-177](#).

Return to the [Summary Table](#).

#### Interrupt Group 9 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-157. PIEIFR9 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-177. PIEIFR9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 9.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 9.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 9.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 9.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 9.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 9.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 9.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 9.1 Reset type: SYSRSn

### 3.15.10.21 PIEIER10 Register (Offset = 14h) [Reset = 0000h]

PIEIER10 is shown in [Figure 3-158](#) and described in [Table 3-178](#).

Return to the [Summary Table](#).

#### Interrupt Group 10 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-158. PIEIER10 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-178. PIEIER10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 10.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 10.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 10.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 10.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 10.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 10.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 10.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 10.1 Reset type: SYSRSn

### 3.15.10.22 PIEIFR10 Register (Offset = 15h) [Reset = 0000h]

PIEIFR10 is shown in [Figure 3-159](#) and described in [Table 3-179](#).

Return to the [Summary Table](#).

#### Interrupt Group 10 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-159. PIEIFR10 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-179. PIEIFR10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 10.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 10.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 10.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 10.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 10.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 10.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 10.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 10.1 Reset type: SYSRSn

### 3.15.10.23 PIEIER11 Register (Offset = 16h) [Reset = 0000h]

PIEIER11 is shown in [Figure 3-160](#) and described in [Table 3-180](#).

Return to the [Summary Table](#).

#### Interrupt Group 11 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-160. PIEIER11 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-180. PIEIER11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 11.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 11.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 11.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 11.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 11.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 11.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 11.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 11.1 Reset type: SYSRSn

### 3.15.10.24 PIEIFR11 Register (Offset = 17h) [Reset = 0000h]

PIEIFR11 is shown in [Figure 3-161](#) and described in [Table 3-181](#).

Return to the [Summary Table](#).

#### Interrupt Group 11 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-161. PIEIFR11 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-181. PIEIFR11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 11.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 11.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 11.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 11.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 11.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 11.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 11.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 11.1 Reset type: SYSRSn

### 3.15.10.25 PIEIER12 Register (Offset = 18h) [Reset = 0000h]

PIEIER12 is shown in [Figure 3-162](#) and described in [Table 3-182](#).

Return to the [Summary Table](#).

#### Interrupt Group 12 Enable Register

These register bits individually enable an interrupt within a group. They behave very much like the bits in the CPU interrupt enable register (IER).

Setting a bit to 1 allows the corresponding interrupt to propagate to the CPU.

Setting a bit to 0 prevents the corresponding interrupt from propagating. Note that a peripheral interrupt signal can still set the PIEIFR bit for the disabled interrupt.

**Figure 3-162. PIEIER12 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-182. PIEIER12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Enable for Interrupt 12.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Enable for Interrupt 12.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Enable for Interrupt 12.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Enable for Interrupt 12.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Enable for Interrupt 12.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Enable for Interrupt 12.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Enable for Interrupt 12.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Enable for Interrupt 12.1 Reset type: SYSRSn

### 3.15.10.26 PIEIFR12 Register (Offset = 19h) [Reset = 0000h]

PIEIFR12 is shown in [Figure 3-163](#) and described in [Table 3-183](#).

Return to the [Summary Table](#).

#### Interrupt Group 12 Flag Register

These register bits indicate whether each interrupt in the group is currently pending. They behave very much like the bits in the CPU interrupt flag register (IFR).

When a peripheral sends an interrupt, the corresponding bit is set. This bit is cleared when the interrupt propagates to the CPU, at which point PIEACK is set.

NOTE: PIE IFR flags can be written to create software interrupts.

The IFR flag will be cleared on a write of zero. Hence, when the intent is to fire an interrupt it may cause inadvertent cancellation of other interrupts. It is recommended to use this only for testing or with extreme caution in the application code. Reading the PIE IFR registers is safe.

**Figure 3-163. PIEIFR12 Register**

15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
INTx8	INTx7	INTx6	INTx5	INTx4	INTx3	INTx2	INTx1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-183. PIEIFR12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R/W	0h	Reserved
7	INTx8	R/W	0h	Flag for Interrupt 12.8 Reset type: SYSRSn
6	INTx7	R/W	0h	Flag for Interrupt 12.7 Reset type: SYSRSn
5	INTx6	R/W	0h	Flag for Interrupt 12.6 Reset type: SYSRSn
4	INTx5	R/W	0h	Flag for Interrupt 12.5 Reset type: SYSRSn
3	INTx4	R/W	0h	Flag for Interrupt 12.4 Reset type: SYSRSn
2	INTx3	R/W	0h	Flag for Interrupt 12.3 Reset type: SYSRSn
1	INTx2	R/W	0h	Flag for Interrupt 12.2 Reset type: SYSRSn
0	INTx1	R/W	0h	Flag for Interrupt 12.1 Reset type: SYSRSn

### 3.15.11 SYNC\_SOC\_REGS Registers

Table 3-184 lists the memory-mapped registers for the SYNC\_SOC\_REGS registers. All register offset addresses not listed in Table 3-184 should be considered as reserved locations and the register contents should not be modified.

**Table 3-184. SYNC\_SOC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SYNCSELECT	Sync Input and Output Select Register	EALLOW	<a href="#">Go</a>
2h	ADCSOCOUTSELECT	External ADCSOC Select Register	EALLOW	<a href="#">Go</a>
4h	SYNCSOCLOCK	SYNCSEL and EXTADCSOC Select Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-185 shows the codes that are used for access types in this section.

**Table 3-185. SYNC\_SOC\_REGS Access Type Codes**

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

### 3.15.11.1 SYNCSELECT Register (Offset = 0h) [Reset = E003FFFFh]

SYNCSELECT is shown in [Figure 3-164](#) and described in [Table 3-186](#).

Return to the [Summary Table](#).

Sync Input and Output Select Register

**Figure 3-164. SYNCSELECT Register**

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

**Table 3-186. SYNCSELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	7h	Reserved

**Table 3-186. SYNCSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28-24	SYNCOUT	R/W	0h	Select Syncout Source: 00000: EPWM1SYNCOUT selected to drive the SYNCOUT pin. 00001: EPWM2SYNCOUT selected to drive the SYNCOUT pin. 00010: EPWM3SYNCOUT selected to drive the SYNCOUT pin. 00011: EPWM4SYNCOUT selected to drive the SYNCOUT pin. 00100: EPWM5SYNCOUT selected to drive the SYNCOUT pin. 00101: EPWM6SYNCOUT selected to drive the SYNCOUT pin. 00110: EPWM7SYNCOUT selected to drive the SYNCOUT pin. 00111: Reserved 01000: Reserved 01001: Reserved 01010: Reserved 01011: Reserved 01100: Reserved 01101: Reserved 01110: Reserved 01111: Reserved 10000: Reserved 10001: Reserved 10010: Reserved 10011: Reserved 10100: Reserved 10101: Reserved 10110: Reserved 10111: Reserved 11000: ECAP1SYNCOUT selected to drive the SYNCOUT pin. 11001: ECAP2SYNCOUT selected to drive the SYNCOUT pin. 11010: Reserved 11011: Reserved 11100: Reserved 11101: Reserved 11110: Reserved 11111: Reserved Notes: [1] Reserved position defaults to 00 selection Reset type: SYSRSn
23-18	RESERVED	R-0	0h	Reserved
17-15	RESERVED	R/W	7h	Reserved
14-12	RESERVED	R/W	7h	Reserved
11-9	RESERVED	R/W	7h	Reserved
8-6	RESERVED	R/W	7h	Reserved
5-3	RESERVED	R/W	7h	Reserved
2-0	RESERVED	R/W	7h	Reserved

### 3.15.11.2 ADCSOCOUTSELECT Register (Offset = 2h) [Reset = 0000000h]

ADCSOCOUTSELECT is shown in [Figure 3-165](#) and described in [Table 3-187](#).

Return to the [Summary Table](#).

External ADCSOC Select Register

**Figure 3-165. ADCSOCOUTSELECT Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	PWM7SOCBEN	PWM6SOCBEN	PWM5SOCBEN	PWM4SOCBEN	PWM3SOCBEN	PWM2SOCBEN	PWM1SOCBEN
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED				RESERVED	RESERVED	RESERVED	RESERVED
R-0-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	PWM7SOCAEN	PWM6SOCAEN	PWM5SOCAEN	PWM4SOCAEN	PWM3SOCAEN	PWM2SOCAEN	PWM1SOCAEN
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 3-187. ADCSOCOUTSELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R-0	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	PWM7SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
21	PWM6SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
20	PWM5SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
19	PWM4SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
18	PWM3SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
17	PWM2SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn

**Table 3-187. ADCSOCOUTSELECT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	PWM1SOCBEN	R/W	0h	ADCSOCBOn source select: 0: Respective EPWM SOCB output is not selected 1: Respective EPWM SOCB output is selected Reset type: SYSRSn
15-12	RESERVED	R-0	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	PWM7SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
5	PWM6SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
4	PWM5SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
3	PWM4SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
2	PWM3SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
1	PWM2SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn
0	PWM1SOCAEN	R/W	0h	ADCSOCAOn source select: 0: Respective EPWM SOCA output is not selected 1: Respective EPWM SOCA output is selected Reset type: SYSRSn

### 3.15.11.3 SYNCSOCLOCK Register (Offset = 4h) [Reset = 0000000h]

SYNCSOCLOCK is shown in [Figure 3-166](#) and described in [Table 3-188](#).

Return to the [Summary Table](#).

SYNCSEL and EXTADCSOC Select Lock register

**Figure 3-166. SYNCSOCLOCK Register**

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

**Table 3-188. SYNCSOCLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1	ADCSOCOUTSELECT	R/WOnce	0h	ADCSOCOUTSELECT Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any bit in this register, once set can only be created through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn
0	SYNCSELECT	R/WOnce	0h	SYNCSELECT Register Lock bit: 0: Respective register is not locked 1: Respective register is locked. Notes: [1] Any bit in this register, once set can only be created through a SYSRSn. Write of 0 to any bit of this register has no effect [2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed Reset type: SYSRSn

### 3.15.12 SYS\_STATUS\_REGS Registers

Table 3-189 lists the memory-mapped registers for the SYS\_STATUS\_REGS registers. All register offset addresses not listed in Table 3-189 should be considered as reserved locations and the register contents should not be modified.

**Table 3-189. SYS\_STATUS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
10h	SYS_ERR_INT_FLG	Status of interrupts due to multiple different errors in the system.		<a href="#">Go</a>
12h	SYS_ERR_INT_CLR	SYS_ERR_INT_FLG clear register		<a href="#">Go</a>
14h	SYS_ERR_INT_SET	SYS_ERR_INT_FLG set register	EALLOW	<a href="#">Go</a>
16h	SYS_ERR_MASK	SYS_ERR_MASK register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-190 shows the codes that are used for access types in this section.

**Table 3-190. SYS\_STATUS\_REGS Access Type Codes**

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

### 3.15.12.1 SYS\_ERR\_INT\_FLG Register (Offset = 10h) [Reset = 0000000h]

SYS\_ERR\_INT\_FLG is shown in [Figure 3-167](#) and described in [Table 3-191](#).

Return to the [Summary Table](#).

Status of interrupts due to multiple different errors in the system.

**Figure 3-167. SYS\_ERR\_INT\_FLG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	RESERVED	FPU_OFLOW	FPU_UFLOW
R-0h				R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	EPG1_INT	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RAM_ACC_VIO L	RESERVED	CORRECTABL E_ERR	RESERVED	GINT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 3-191. SYS\_ERR\_INT\_FLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
17	FPU_OFLOW	R	0h	0: FPU_OFLOW has not fired an interrupt. 1: FPU_OFLOW has fired an interrupt Reset type: SYSRSn
16	FPU_UFLOW	R	0h	0: FPU_UFLOW has not fired an interrupt. 1: FPU_UFLOW has fired an interrupt Reset type: SYSRSn
15	RESERVED	R	0h	Reserved
14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	RESERVED	R	0h	Reserved
11	EPG1_INT	R	0h	0: EPG1_INT has not fired an interrupt. 1: EPG1_INT has fired an interrupt Reset type: SYSRSn
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RAM_ACC_VIO L	R	0h	0: None of the Masters have violated the set protection rules 1: At least one of the master accesses has violated one or more of the access protection rules Reset type: SYSRSn
3	RESERVED	R	0h	Reserved

**Table 3-191. SYS\_ERR\_INT\_FLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	CORRECTABLE_ERR	R	0h	0: Number of correctable errors detected has not exceeded the set threshold for flash/RAM. 1: Number of correctable errors detected has exceeded the set threshold for flash/RAM. Reset type: SYSRSn
1	RESERVED	R	0h	Reserved
0	GINT	R	0h	Global Interrupt flag: 0: On any of the flags of SYS_ERR_INT_FLG register being set, SYS_ERR_INT is pulsed and GINT flag would be set 1: No further interrupts would be fired until GINT flag is cleared Reset type: SYSRSn

### 3.15.12.2 SYS\_ERR\_INT\_CLR Register (Offset = 12h) [Reset = 0000000h]

SYS\_ERR\_INT\_CLR is shown in [Figure 3-168](#) and described in [Table 3-192](#).

Return to the [Summary Table](#).

SYS\_ERR\_INT\_FLG clear register

**Figure 3-168. SYS\_ERR\_INT\_CLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	RESERVED	FPU_OFLOW	FPU_UFLOW
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	EPG1_INT	RESERVED	RESERVED	RESERVED
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RAM_ACC_VIO L	RESERVED	CORRECTABL E_ERR	RESERVED	GINT
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 3-192. SYS\_ERR\_INT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19	RESERVED	R-0/W1S	0h	Reserved
18	RESERVED	R-0/W1S	0h	Reserved
17	FPU_OFLOW	R-0/W1S	0h	0: No effect 1: FPU_OFLOW flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn
16	FPU_UFLOW	R-0/W1S	0h	0: No effect 1: FPU_UFLOW flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn
15	RESERVED	R-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	EPG1_INT	R-0/W1S	0h	0: No effect 1: EPG1_INT flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RAM_ACC_VIOL	R-0/W1S	0h	0: No effect 1: RAM_ACC_VIOL flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn
3	RESERVED	R-0/W1S	0h	Reserved

**Table 3-192. SYS\_ERR\_INT\_CLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	CORRECTABLE_ERR	R-0/W1S	0h	0: No effect 1: CORRECTABLE_ERR flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn
1	RESERVED	R-0/W1S	0h	Reserved
0	GINT	R-0/W1S	0h	0: No effect 1: GINT flag of SYS_ERR_INT_FLG register will be cleared. Reset type: SYSRSn

### 3.15.12.3 SYS\_ERR\_INT\_SET Register (Offset = 14h) [Reset = 0000000h]

SYS\_ERR\_INT\_SET is shown in [Figure 3-169](#) and described in [Table 3-193](#).

Return to the [Summary Table](#).

SYS\_ERR\_INT\_FLG set register

**Figure 3-169. SYS\_ERR\_INT\_SET Register**

31	30	29	28	27	26	25	24
KEY							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	RESERVED	FPU_OFLOW	FPU_UFLOW
R-0h				R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	EPG1_INT	RESERVED	RESERVED	RESERVED
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RAM_ACC_VIO L	RESERVED	CORRECTABL E_ERR	RESERVED	RESERVED
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h

**Table 3-193. SYS\_ERR\_INT\_SET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	KEY	R-0/W	0h	A value of 0xa5 to this field would enable write to the other bit fields of this register. Any other value written to KEY field would block the write to the other fields of this register. Note: Only a 32 bit write to this register will succeed in updating the fields of this register, provided the correct value written to the KEY field simultaneously Reset type: SYSRSn
23-20	RESERVED	R	0h	Reserved
19	RESERVED	R-0/W1S	0h	Reserved
18	RESERVED	R-0/W1S	0h	Reserved
17	FPU_OFLOW	R-0/W1S	0h	0: No effect 1: FPU_OFLOW flag of SYS_ERR_INT_FLG register will be set. Reset type: SYSRSn
16	FPU_UFLOW	R-0/W1S	0h	0: No effect 1: FPU_UFLOW flag of SYS_ERR_INT_FLG register will be set. Reset type: SYSRSn
15	RESERVED	R-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	EPG1_INT	R-0/W1S	0h	0: No effect 1: EPG1_INT flag of SYS_ERR_INT_FLG register will be set. Reset type: SYSRSn
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved
6	RESERVED	R-0/W1S	0h	Reserved

**Table 3-193. SYS\_ERR\_INT\_SET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	RESERVED	R-0/W1S	0h	Reserved
4	RAM_ACC_VIOL	R-0/W1S	0h	0: No effect 1: RAM_ACC_VIOL flag of SYS_ERR_INT_FLG register will be set. Reset type: SYSRSn
3	RESERVED	R-0/W1S	0h	Reserved
2	CORRECTABLE_ERR	R-0/W1S	0h	0: No effect 1: CORRECTABLE_ERR flag of SYS_ERR_INT_FLG register will be set. Reset type: SYSRSn
1	RESERVED	R-0/W1S	0h	Reserved
0	RESERVED	R	0h	Reserved

### 3.15.12.4 SYS\_ERR\_MASK Register (Offset = 16h) [Reset = 0000000h]

SYS\_ERR\_MASK is shown in [Figure 3-170](#) and described in [Table 3-194](#).

Return to the [Summary Table](#).

SYS\_ERR\_MASK register

**Figure 3-170. SYS\_ERR\_MASK Register**

31	30	29	28	27	26	25	24
KEY							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED				RESERVED	RESERVED	FPU_OFLOW	FPU_UFLOW
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	EPG1_INT	RESERVED	RESERVED	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
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RAM_ACC_VIO L	RESERVED	CORRECTABL E_ERR	RESERVED	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 3-194. SYS\_ERR\_MASK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	KEY	R/W	0h	A value of 0xa5 to this field would enable write to the other bit fields of this register. Any other value written to KEY field would block the write to the other fields of this register. Note: Only a 32 bit write to this register will succeed in updating the fields of this register, provided the correct value written to the KEY field simultaneously Reset type: SYSRSn
23-20	RESERVED	R	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	FPU_OFLOW	R/W	0h	0: FPU_OFLOW flag of SYS_ERR_INT_FLG register will be set on a hardware event. 1: FPU_OFLOW flag of SYS_ERR_INT_FLG register will not be set on a hardware event. Reset type: SYSRSn
16	FPU_UFLOW	R/W	0h	0: FPU_UFLOW flag of SYS_ERR_INT_FLG register will be set on a hardware event. 1: FPU_UFLOW flag of SYS_ERR_INT_FLG register will not be set on a hardware event. Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	EPG1_INT	R/W	0h	0: EPG1_INT flag of SYS_ERR_INT_FLG register will be set on a hardware event. 1: EPG1_INT flag of SYS_ERR_INT_FLG register will not be set on a hardware event. Reset type: SYSRSn

**Table 3-194. SYS\_ERR\_MASK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RAM_ACC_VIOL	R/W	0h	0: RAM_ACC_VIOL flag of SYS_ERR_INT_FLG reister will be set on a hardware event. 1: RAM_ACC_VIOL flag of SYS_ERR_INT_FLG reister will not be set on a hardware event. Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	CORRECTABLE_ERR	R/W	0h	0: CORRECTABLE_ERR flag of SYS_ERR_INT_FLG reister will be set on a hardware event. 1: CORRECTABLE_ERR flag of SYS_ERR_INT_FLG reister will not be set on a hardware event. Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R	0h	Reserved

### 3.15.13 TEST\_ERROR\_REGS Registers

Table 3-195 lists the memory-mapped registers for the TEST\_ERROR\_REGS registers. All register offset addresses not listed in Table 3-195 should be considered as reserved locations and the register contents should not be modified.

**Table 3-195. TEST\_ERROR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	CPU_RAM_TEST_ERROR_STS	Ram Test: Error Status Register		<a href="#">Go</a>
2h	CPU_RAM_TEST_ERROR_STS_CLR	Ram Test: Error Status Clear Register		<a href="#">Go</a>
4h	CPU_RAM_TEST_ERROR_ADDR	Ram Test: Error address register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-196 shows the codes that are used for access types in this section.

**Table 3-196. TEST\_ERROR\_REGS Access Type Codes**

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

### 3.15.13.1 CPU\_RAM\_TEST\_ERROR\_STS Register (Offset = 0h) [Reset = 0000000h]

CPU\_RAM\_TEST\_ERROR\_STS is shown in [Figure 3-171](#) and described in [Table 3-197](#).

Return to the [Summary Table](#).

Ram Test: Error Status Register

**Figure 3-171. CPU\_RAM\_TEST\_ERROR\_STS 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_ERROR	COR_ERROR
R-0h						R-0h	R-0h

**Table 3-197. CPU\_RAM\_TEST\_ERROR\_STS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	UNC_ERROR	R	0h	0: Indicates that there were no 'un-correctable errors' generated in the RAM/ROM test mode. 1: Indicates that 'un-correctable errors' wer generated in the RAM/ROM test mode. Reset type: SYSRSn
0	COR_ERROR	R	0h	0: Indicates that there were no 'correctable errors' generated in the RAM/ROM test mode. 1: Indicates that 'correctable errors' wer generated in the RAM/ROM test mode. Reset type: SYSRSn

### 3.15.13.2 CPU\_RAM\_TEST\_ERROR\_STS\_CLR Register (Offset = 2h) [Reset = 0000000h]

CPU\_RAM\_TEST\_ERROR\_STS\_CLR is shown in [Figure 3-172](#) and described in [Table 3-198](#).

Return to the [Summary Table](#).

Ram Test: Error Status Clear Register

**Figure 3-172. CPU\_RAM\_TEST\_ERROR\_STS\_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						UNC_ERROR	COR_ERROR
R-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 3-198. CPU\_RAM\_TEST\_ERROR\_STS\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	UNC_ERROR	R-0/W1S	0h	0: No effect. 1: Clears the corresponding bit in CPU_RAM_TEST_ERROR_STS register. Reset type: SYSRSn
0	COR_ERROR	R-0/W1S	0h	0: No effect. 1: Clears the corresponding bit in CPU_RAM_TEST_ERROR_STS register. Reset type: SYSRSn

### 3.15.13.3 CPU\_RAM\_TEST\_ERROR\_ADDR Register (Offset = 4h) [Reset = 0000000h]

CPU\_RAM\_TEST\_ERROR\_ADDR is shown in [Figure 3-173](#) and described in [Table 3-199](#).

Return to the [Summary Table](#).

Ram Test: Error address register

**Figure 3-173. CPU\_RAM\_TEST\_ERROR\_ADDR Register**

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

**Table 3-199. CPU\_RAM\_TEST\_ERROR\_ADDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R	0h	Address of the location where error was detected in RAM/ROM test modes. Reset type: SYSRSn

### 3.15.14 UID\_REGS Registers

Table 3-200 lists the memory-mapped registers for the UID\_REGS registers. All register offset addresses not listed in Table 3-200 should be considered as reserved locations and the register contents should not be modified.

**Table 3-200. UID\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	UID_PSRAND0	UID Psuedo-random 160 bit number		<a href="#">Go</a>
2h	UID_PSRAND1	UID Psuedo-random 160 bit number		<a href="#">Go</a>
4h	UID_PSRAND2	UID Psuedo-random 160 bit number		<a href="#">Go</a>
6h	UID_PSRAND3	UID Psuedo-random 160 bit number		<a href="#">Go</a>
8h	UID_PSRAND4	UID Psuedo-random 160 bit number		<a href="#">Go</a>
Ah	UID_UNIQUE0	UID Unique 64 bit number		<a href="#">Go</a>
Ch	UID_UNIQUE1	UID Unique 64 bit number		<a href="#">Go</a>
Eh	UID_CHECKSUM	UID Checksum		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-201 shows the codes that are used for access types in this section.

**Table 3-201. UID\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
Reset or Default Value		
-n		Value after reset or the default value

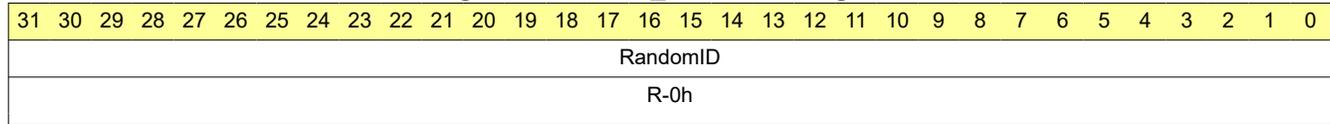
### 3.15.14.1 UID\_PSRAND0 Register (Offset = 0h) [Reset = 00000000h]

UID\_PSRAND0 is shown in [Figure 3-174](#) and described in [Table 3-202](#).

Return to the [Summary Table](#).

UID Psuedo-random 160 bit number

**Figure 3-174. UID\_PSRAND0 Register**



**Table 3-202. UID\_PSRAND0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	0h	Psuedorandom portion of the UID Reset type: N/A

### 3.15.14.2 UID\_PSRAND1 Register (Offset = 2h) [Reset = 0000000h]

UID\_PSRAND1 is shown in [Figure 3-175](#) and described in [Table 3-203](#).

Return to the [Summary Table](#).

UID Psuedo-random 160 bit number

**Figure 3-175. UID\_PSRAND1 Register**

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

**Table 3-203. UID\_PSRAND1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	0h	Psuedorandom portion of the UID Reset type: N/A

### 3.15.14.3 UID\_PSRAND2 Register (Offset = 4h) [Reset = 00000000h]

UID\_PSRAND2 is shown in [Figure 3-176](#) and described in [Table 3-204](#).

Return to the [Summary Table](#).

UID Psuedo-random 160 bit number

**Figure 3-176. UID\_PSRAND2 Register**

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

**Table 3-204. UID\_PSRAND2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	0h	Psuedorandom portion of the UID Reset type: N/A

### 3.15.14.4 UID\_PSRAND3 Register (Offset = 6h) [Reset = 0000000h]

UID\_PSRAND3 is shown in [Figure 3-177](#) and described in [Table 3-205](#).

Return to the [Summary Table](#).

UID Psuedo-random 160 bit number

**Figure 3-177. UID\_PSRAND3 Register**

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

**Table 3-205. UID\_PSRAND3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	0h	Psuedorandom portion of the UID Reset type: N/A

### 3.15.14.5 UID\_PSRAND4 Register (Offset = 8h) [Reset = 0000000h]

UID\_PSRAND4 is shown in [Figure 3-178](#) and described in [Table 3-206](#).

Return to the [Summary Table](#).

UID Psuedo-random 160 bit number

**Figure 3-178. UID\_PSRAND4 Register**

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

**Table 3-206. UID\_PSRAND4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RandomID	R	0h	Psuedorandom portion of the UID Reset type: N/A

### 3.15.14.6 UID\_UNIQUE0 Register (Offset = Ah) [Reset = 0000000h]

UID\_UNIQUE0 is shown in [Figure 3-179](#) and described in [Table 3-207](#).

Return to the [Summary Table](#).

UID Unique 64 bit number

**Figure 3-179. UID\_UNIQUE0 Register**

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

**Table 3-207. UID\_UNIQUE0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UniqueID	R	0h	Unique portion of the UID. This identifier will be unique across all devices with the same PARTIDH. Reset type: N/A

### 3.15.14.7 UID\_UNIQUE1 Register (Offset = Ch) [Reset = 0000000h]

UID\_UNIQUE1 is shown in [Figure 3-180](#) and described in [Table 3-208](#).

Return to the [Summary Table](#).

UID Unique 64 bit number

**Figure 3-180. UID\_UNIQUE1 Register**

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

**Table 3-208. UID\_UNIQUE1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	UniqueID	R	0h	Unique portion of the UID. This identifier will be unique across all devices with the same PARTIDH. Reset type: N/A

### 3.15.14.8 UID\_CHECKSUM Register (Offset = Eh) [Reset = 0000000h]

UID\_CHECKSUM is shown in [Figure 3-181](#) and described in [Table 3-209](#).

Return to the [Summary Table](#).

Fletcher checksum of UID\_PSRAND and UID\_UNIQUE registers

**Figure 3-181. UID\_CHECKSUM Register**

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

**Table 3-209. UID\_CHECKSUM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Checksum	R	0h	Fletcher checksum of UID_PSRANDx and UID_UNIQUE Reset type: N/A

### 3.15.15 WD\_REGS Registers

Table 3-210 lists the memory-mapped registers for the WD\_REGS registers. All register offset addresses not listed in Table 3-210 should be considered as reserved locations and the register contents should not be modified.

**Table 3-210. WD\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
22h	SCSR	System Control & Status Register	EALLOW	<a href="#">Go</a>
23h	WDCNTR	Watchdog Counter Register	EALLOW	<a href="#">Go</a>
25h	WDKEY	Watchdog Reset Key Register	EALLOW	<a href="#">Go</a>
29h	WDCR	Watchdog Control Register	EALLOW	<a href="#">Go</a>
2Ah	WDWCR	Watchdog Windowed Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 3-211 shows the codes that are used for access types in this section.

**Table 3-211. WD\_REGS Access Type Codes**

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

### 3.15.15.1 SCSR Register (Offset = 22h) [Reset = 5h]

SCSR is shown in [Figure 3-182](#) and described in [Table 3-212](#).

Return to the [Summary Table](#).

System Control & Status Register

**Figure 3-182. SCSR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					WDINTS	WDENINT	WDOVERRIDE
R-0-0h					R-1h	R/W-0h	R/W1S-1h

**Table 3-212. SCSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	WDINTS	R	1h	Watchdog Interrupt Status This bit indicates the state of the active-low watchdog interrupt signal (synchronized to SYSCLK). If the watchdog interrupt is used to wake the system from a low-power mode, then that mode should only be entered while this bit is high. Likewise, this bit must go high before the watchdog can be safely disabled and re-enabled. Reset type: SYSRSn
1	WDENINT	R/W	0h	Watchdog Interrupt Enable/Reset Disable This bit determines whether the watchdog triggers an interrupt (WAKE/WDOG) or a reset (WDRS) when the counter expires. Reset type: SYSRSn
0	WDOVERRIDE	R/W1S	1h	Watchdog Enable Lock Writing a 1 to this bit clears it and locks the WDDIS bit in the WDCR register. The bit will remain in this state until the next system reset. Reads of this bit return its current value. Writing a 0 to this bit has no effect. Reset type: SYSRSn

### 3.15.15.2 WDCNTR Register (Offset = 23h) [Reset = 0h]

WDCNTR is shown in [Figure 3-183](#) and described in [Table 3-213](#).

Return to the [Summary Table](#).

Watchdog Counter Register

**Figure 3-183. WDCNTR Register**

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

**Table 3-213. WDCNTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-0	WDCNTR	R	0h	Watchdog Counter These bits contain the current value of the watchdog counter. This counter increments with each WDCLK (INTOSC1) cycle. If the counter overflows, either an interrupt or a reset is generated based on the value of the WDINTEN bit in the SCSR register. If the correct value is written to the WDKEY register, this counter is reset to zero. Reset type: IORSn

### 3.15.15.3 WDKEY Register (Offset = 25h) [Reset = 0h]

WDKEY is shown in [Figure 3-184](#) and described in [Table 3-214](#).

Return to the [Summary Table](#).

Watchdog Reset Key Register

**Figure 3-184. WDKEY Register**

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

**Table 3-214. WDKEY Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-0	WDKEY	R/W	0h	Watchdog Counter Reset Writing 0x55 followed by 0xAA will cause the watchdog counter to reset to zero, preventing an overflow. Writing other values has no effect. Reads of this register return the value of the WDCR register. Reset type: IORSn

### 3.15.15.4 WDCR Register (Offset = 29h) [Reset = 0h]

WDCR is shown in [Figure 3-185](#) and described in [Table 3-215](#).

Return to the [Summary Table](#).

#### Watchdog Control Register

This memory mapped register requires a delay of 45 SYSCLK cycles between subsequent writes to the register, otherwise a second write can be lost. This delay can be realized by adding 45 NOP instructions.

**Figure 3-185. WDCR Register**

15	14	13	12	11	10	9	8
RESERVED				WDPRECLKDIV			
R-0-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	WDDIS	WDCHK		WDPS			
R/W1S-0h	R/W-0h	R-0/W-0h		R/W-0h			

**Table 3-215. WDCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-8	WDPRECLKDIV	R/W	0h	Watchdog Clock Pre-divider These bits determine the watchdog clock pre-divider, which is the first of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas: PREDIVCLK = INTOSC1 / Pre-divider WDCLK = PREDIVCLK / Prescaler The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default pre-divider value is 512. Reset type: IORSn
7	RESERVED	R/W1S	0h	Reserved
6	WDDIS	R/W	0h	Watchdog Disable Setting this bit disables the watchdog module. Clearing this bit enables the watchdog module. This bit can be locked by the WDOVERRIDE bit in the SCSR register. The watchdog is enabled on reset. Reset type: IORSn
5-3	WDCHK	R-0/W	0h	Watchdog Check Bits During any write to this register, these bits must be written with the value 101 (binary). Writing any other value will immediately trigger the watchdog reset or interrupt. Reset type: IORSn
2-0	WDPS	R/W	0h	Watchdog Clock Prescaler These bits determine the watchdog clock prescaler, which is the second of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas: PREDIVCLK = INTOSC1 / Pre-divider WDCLK = PREDIVCLK / Prescaler The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default prescaler value is 1. Reset type: IORSn

### 3.15.15.5 WDWCR Register (Offset = 2Ah) [Reset = 0h]

WDWCR is shown in [Figure 3-186](#) and described in [Table 3-216](#).

Return to the [Summary Table](#).

Watchdog Windowed Control Register

**Figure 3-186. WDWCR Register**

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

**Table 3-216. WDWCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R-0	0h	Reserved
8	RESERVED	R	0h	Reserved
7-0	MIN	R/W	0h	Watchdog Window Threshold These bits specify the lower limit of the watchdog counter reset window. If the counter is reset via the WDKEY register before the counter value reaches the value in this register, the watchdog immediately triggers a reset or interrupt. Reset type: IORSn

### 3.15.16 XINT\_REGS Registers

Table 3-217 lists the memory-mapped registers for the XINT\_REGS registers. All register offset addresses not listed in Table 3-217 should be considered as reserved locations and the register contents should not be modified.

**Table 3-217. XINT\_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-218 shows the codes that are used for access types in this section.

**Table 3-218. XINT\_REGS Access Type Codes**

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

### 3.15.16.1 XINT1CR Register (Offset = 0h) [Reset = 0000h]

XINT1CR is shown in [Figure 3-187](#) and described in [Table 3-219](#).

Return to the [Summary Table](#).

XINT1 configuration register

**Figure 3-187. XINT1CR Register**

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

**Table 3-219. XINT1CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.15.16.2 XINT2CR Register (Offset = 1h) [Reset = 0000h]

XINT2CR is shown in [Figure 3-188](#) and described in [Table 3-220](#).

Return to the [Summary Table](#).

XINT2 configuration register

**Figure 3-188. XINT2CR Register**

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

**Table 3-220. XINT2CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.15.16.3 XINT3CR Register (Offset = 2h) [Reset = 0000h]

XINT3CR is shown in [Figure 3-189](#) and described in [Table 3-221](#).

Return to the [Summary Table](#).

XINT3 configuration register

**Figure 3-189. XINT3CR Register**

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

**Table 3-221. XINT3CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.15.16.4 XINT4CR Register (Offset = 3h) [Reset = 0000h]

XINT4CR is shown in [Figure 3-190](#) and described in [Table 3-222](#).

Return to the [Summary Table](#).

XINT4 configuration register

**Figure 3-190. XINT4CR Register**

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

**Table 3-222. XINT4CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.15.16.5 XINT5CR Register (Offset = 4h) [Reset = 0000h]

XINT5CR is shown in [Figure 3-191](#) and described in [Table 3-223](#).

Return to the [Summary Table](#).

XINT5 configuration register

**Figure 3-191. XINT5CR Register**

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

**Table 3-223. XINT5CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3-2	POLARITY	R/W	0h	00: Interrupt is selected as negative edge triggered 01: Interrupt is selected as positive edge triggered 10: Interrupt is selected as negative edge triggered 11: Interrupt is selected as positive or negative edge triggered Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	0: Interrupt Disabled 1: Interrupt Enabled Reset type: SYSRSn

### 3.15.16.6 XINT1CTR Register (Offset = 8h) [Reset = 0000h]

XINT1CTR is shown in [Figure 3-192](#) and described in [Table 3-224](#).

Return to the [Summary Table](#).

XINT1 counter register

**Figure 3-192. XINT1CTR Register**

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

**Table 3-224. XINT1CTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	INTCTR	R	0h	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset. Reset type: SYSRSn

### 3.15.16.7 XINT2CTR Register (Offset = 9h) [Reset = 0000h]

XINT2CTR is shown in [Figure 3-193](#) and described in [Table 3-225](#).

Return to the [Summary Table](#).

XINT2 counter register

**Figure 3-193. XINT2CTR Register**

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

**Table 3-225. 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. Reset type: SYSRSn

### 3.15.16.8 XINT3CTR Register (Offset = Ah) [Reset = 0000h]

XINT3CTR is shown in [Figure 3-194](#) and described in [Table 3-226](#).

Return to the [Summary Table](#).

XINT3 counter register

**Figure 3-194. XINT3CTR Register**

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

**Table 3-226. XINT3CTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	INTCTR	R	0h	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. The counter must only be reset by the selected POLARITY edge as selected in the respective interrupt control register. When the interrupt is disabled, the counter will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset. Reset type: SYSRSn

### 3.15.17 Register to Driverlib Function Mapping

#### 3.15.17.1 ASYSCTL Registers to Driverlib Functions

**Table 3-227. ASYSCTL Registers to Driverlib Functions**

File	Driverlib Function
<b>EXTROSCSR1</b>	
asysctl.h	ASysCtl_getExtROscStatus
sysctl.c	SysCtl_setClock
<b>INTERNALTESTCTL</b>	
asysctl.h	ASysCtl_selectInternalTestNode
<b>CONFIGLOCK</b>	
-	
<b>TSNSCTL</b>	
asysctl.h	ASysCtl_enableTemperatureSensor
asysctl.h	ASysCtl_disableTemperatureSensor
<b>ANAREFCTL</b>	
adc.c	ADC_setVREF
adc.c	ADC_setOffsetTrim
asysctl.h	ASysCtl_setAnalogReferenceInternal
asysctl.h	ASysCtl_setAnalogReferenceExternal
asysctl.h	ASysCtl_setAnalogReference2P5
asysctl.h	ASysCtl_setAnalogReference1P65
<b>VMONCTL</b>	
-	
<b>CMPPMXSEL</b>	
asysctl.h	ASysCtl_selectCMPPMux
<b>CMPLPMXSEL</b>	
asysctl.h	ASysCtl_selectCMPLPMux
<b>CMPHNMXSEL</b>	
asysctl.h	ASysCtl_selectCMPHNMux
asysctl.h	ASysCtl_selectCMPHNMuxValue
<b>CMPLNMXSEL</b>	
asysctl.h	ASysCtl_selectCMPLNMux
asysctl.h	ASysCtl_selectCMPLNMuxValue
<b>ADCDACLOOPBACK</b>	
asysctl.h	ASysCtl_enableADCDACTLoopback
asysctl.h	ASysCtl_disableADCDACTLoopback
<b>CMPSSCTL</b>	
asysctl.h	ASysCtl_enableCMPSSExternalIDAC
asysctl.h	ASysCtl_disableCMPSSExternalIDAC
<b>LOCK</b>	
asysctl.h	ASysCtl_lockTemperatureSensor
asysctl.h	ASysCtl_lockANAREF
asysctl.h	ASysCtl_lockVMON
asysctl.h	ASysCtl_lockCMPPMux
asysctl.h	ASysCtl_lockCMPLPMux
asysctl.h	ASysCtl_lockCMPHNMux

**Table 3-227. ASYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
asysctl.h	ASysCtl_lockCPLNMux
asysctl.h	ASysCtl_lockVREG
asysctl.h	ASysCtl_lockCMPSSCTL
<b>AGPIOCTRLA</b>	
gpio.c	GPIO_setAnalogMode
<b>AGPIOCTRLH</b>	
-	

### 3.15.17.2 CPUTIMER Registers to Driverlib Functions

**Table 3-228. CPUTIMER Registers to Driverlib Functions**

File	Driverlib Function
<b>TIM</b>	
cputimer.h	CPUTimer_getTimerCount
<b>PRD</b>	
cputimer.h	CPUTimer_setPeriod
<b>TCR</b>	
cputimer.c	CPUTimer_setEmulationMode
cputimer.h	CPUTimer_clearOverflowFlag
cputimer.h	CPUTimer_disableInterrupt
cputimer.h	CPUTimer_enableInterrupt
cputimer.h	CPUTimer_reloadTimerCounter
cputimer.h	CPUTimer_stopTimer
cputimer.h	CPUTimer_resumeTimer
cputimer.h	CPUTimer_startTimer
cputimer.h	CPUTimer_getTimerOverflowStatus
<b>TPR</b>	
cputimer.h	CPUTimer_setPreScaler
<b>TPRH</b>	
cputimer.h	CPUTimer_setPreScaler

### 3.15.17.3 MEMCFG Registers to Driverlib Functions

**Table 3-229. MEMCFG Registers to Driverlib Functions**

File	Driverlib Function
<b>DXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>DXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>DXACCPROT0</b>	
memcfg.c	MemCfg_setProtection
<b>DXACCPROT1</b>	
-	
<b>DXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>DXINIT</b>	

**Table 3-229. MEMCFG Registers to Driverlib Functions (continued)**

File	Driverlib Function
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>DXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>DXRAMTEST_LOCK</b>	
memcfg.c	MemCfg_lockTestConfig
memcfg.c	MemCfg_unlockTestConfig
<b>LSXLOCK</b>	
memcfg.c	MemCfg_lockConfig
memcfg.c	MemCfg_unlockConfig
<b>LSXCOMMIT</b>	
memcfg.c	MemCfg_commitConfig
<b>LSXACCPROTO</b>	
memcfg.c	MemCfg_setProtection
<b>LSXTEST</b>	
memcfg.c	MemCfg_setTestMode
<b>LSXINIT</b>	
memcfg.c	MemCfg_initSections
memcfg.c	MemCfg_getInitStatus
<b>LSXINITDONE</b>	
memcfg.c	MemCfg_getInitStatus
<b>LSXRAMTEST_LOCK</b>	
memcfg.c	MemCfg_lockTestConfig
memcfg.c	MemCfg_unlockTestConfig
<b>ROM_LOCK</b>	
memcfg.c	MemCfg_lockTestConfig
memcfg.c	MemCfg_unlockTestConfig
<b>ROM_TEST</b>	
memcfg.c	MemCfg_setTestMode
<b>ROM_FORCE_ERROR</b>	
memcfg.c	MemCfg_forceMemError
<b>MAVFLG</b>	
memcfg.h	MemCfg_getViolationInterruptStatus
<b>MAVSET</b>	
memcfg.h	MemCfg_forceViolationInterrupt
<b>MAVCLR</b>	
memcfg.h	MemCfg_clearViolationInterruptStatus
<b>MAVINTEN</b>	
memcfg.h	MemCfg_enableViolationInterrupt
memcfg.h	MemCfg_disableViolationInterrupt
<b>MCPUFAVADDR</b>	
memcfg.c	MemCfg_getViolationAddress
<b>MCPUWRAVADDR</b>	
memcfg.c	MemCfg_getViolationAddress
<b>UCERRFLG</b>	

**Table 3-229. MEMCFG Registers to Driverlib Functions (continued)**

File	Driverlib Function
memcfg.h	MemCfg_getUncorrErrorStatus
<b>UCERRSET</b>	
memcfg.h	MemCfg_forceUncorrErrorStatus
<b>UCERRCLR</b>	
memcfg.h	MemCfg_clearUncorrErrorStatus
<b>UCCPUREADDR</b>	
memcfg.c	MemCfg_getUncorrErrorAddress
<b>FLUCERRSTATUS</b>	
-	
<b>FLCERRSTATUS</b>	
-	
<b>CERRFLG</b>	
memcfg.h	MemCfg_getCorrErrorStatus
<b>CERRSET</b>	
memcfg.h	MemCfg_forceCorrErrorStatus
<b>CERRCLR</b>	
memcfg.h	MemCfg_clearCorrErrorStatus
<b>CCPUREADDR</b>	
memcfg.c	MemCfg_getCorrErrorAddress
<b>CERRCNT</b>	
memcfg.h	MemCfg_getCorrErrorCount
<b>CERRTHRES</b>	
memcfg.h	MemCfg_setCorrErrorThreshold
<b>CEINTFLG</b>	
memcfg.h	MemCfg_getCorrErrorInterruptStatus
<b>CEINTCLR</b>	
memcfg.h	MemCfg_clearCorrErrorInterruptStatus
<b>CEINTSET</b>	
memcfg.h	MemCfg_forceCorrErrorInterrupt
<b>CEINTEN</b>	
memcfg.h	MemCfg_enableCorrErrorInterrupt
memcfg.h	MemCfg_disableCorrErrorInterrupt
<b>CPU_RAM_TEST_ERROR_STS</b>	
memcfg.h	MemCfg_getDiagErrorStatus
memcfg.h	MemCfg_clearDiagErrorStatus
<b>CPU_RAM_TEST_ERROR_STS_CLR</b>	
memcfg.h	MemCfg_clearDiagErrorStatus
<b>CPU_RAM_TEST_ERROR_ADDR</b>	
memcfg.h	MemCfg_getDiagErrorAddress

### 3.15.17.4 NMI Registers to Driverlib Functions

**Table 3-230. NMI Registers to Driverlib Functions**

File	Driverlib Function
<b>CFG</b>	
sysctl.h	SysCtl_enableNMIGlobalInterrupt

**Table 3-230. NMI Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>FLG</b>	
sysctl.h	SysCtl_getNMISStatus
sysctl.h	SysCtl_getNMIFlagStatus
sysctl.h	SysCtl_isNMIFlagSet
sysctl.h	SysCtl_clearNMISStatus
sysctl.h	SysCtl_clearAllNMIFlags
sysctl.h	SysCtl_forceNMIFlags
<b>FLGCLR</b>	
sysctl.h	SysCtl_clearNMISStatus
sysctl.h	SysCtl_clearAllNMIFlags
<b>FLGFRC</b>	
sysctl.h	SysCtl_forceNMIFlags
<b>WDCNT</b>	
sysctl.h	SysCtl_getNMIWatchdogCounter
<b>WDPRD</b>	
sysctl.h	SysCtl_setNMIWatchdogPeriod
sysctl.h	SysCtl_getNMIWatchdogPeriod
<b>SHDFLG</b>	
sysctl.h	SysCtl_getNMIShadowFlagStatus
sysctl.h	SysCtl_isNMIShadowFlagSet
<b>ERRORSTS</b>	
sysctl.h	SysCtl_isErrorTriggered
sysctl.h	SysCtl_getErrorPinStatus
sysctl.h	SysCtl_forceError
sysctl.h	SysCtl_clearError
<b>ERRORSTSCLR</b>	
sysctl.h	SysCtl_clearError
<b>ERRORSTSFRC</b>	
sysctl.h	SysCtl_forceError
<b>ERRORCTL</b>	
sysctl.h	SysCtl_selectErrPinPolarity
<b>ERRORLOCK</b>	
sysctl.h	SysCtl_lockErrControl

### 3.15.17.5 PIE Registers to Driverlib Functions

**Table 3-231. PIE Registers to Driverlib Functions**

File	Driverlib Function
<b>CTRL</b>	
interrupt.c	Interrupt_initModule
interrupt.c	Interrupt_defaultHandler
interrupt.h	Interrupt_enablePIE
interrupt.h	Interrupt_disablePIE
<b>ACK</b>	
interrupt.c	Interrupt_disable
interrupt.h	Interrupt_clearACKGroup

**Table 3-231. PIE Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>IER1</b>	
interrupt.c	Interrupt_initModule
interrupt.c	Interrupt_enable
interrupt.c	Interrupt_disable
<b>IFR1</b>	
interrupt.c	Interrupt_initModule
<b>IER2</b>	
interrupt.c	Interrupt_initModule
<b>IFR2</b>	
interrupt.c	Interrupt_initModule
<b>IER3</b>	
interrupt.c	Interrupt_initModule
<b>IFR3</b>	
interrupt.c	Interrupt_initModule
<b>IER4</b>	
interrupt.c	Interrupt_initModule
<b>IFR4</b>	
interrupt.c	Interrupt_initModule
<b>IER5</b>	
interrupt.c	Interrupt_initModule
<b>IFR5</b>	
interrupt.c	Interrupt_initModule
<b>IER6</b>	
interrupt.c	Interrupt_initModule
<b>IFR6</b>	
interrupt.c	Interrupt_initModule
<b>IER7</b>	
interrupt.c	Interrupt_initModule
<b>IFR7</b>	
interrupt.c	Interrupt_initModule
<b>IER8</b>	
interrupt.c	Interrupt_initModule
<b>IFR8</b>	
interrupt.c	Interrupt_initModule
<b>IER9</b>	
interrupt.c	Interrupt_initModule
<b>IFR9</b>	
interrupt.c	Interrupt_initModule
<b>IER10</b>	
interrupt.c	Interrupt_initModule
<b>IFR10</b>	
interrupt.c	Interrupt_initModule
<b>IER11</b>	
interrupt.c	Interrupt_initModule
<b>IFR11</b>	

**Table 3-231. PIE Registers to Driverlib Functions (continued)**

File	Driverlib Function
interrupt.c	Interrupt_initModule
<b>IER12</b>	
interrupt.c	Interrupt_initModule
<b>IFR12</b>	
interrupt.c	Interrupt_initModule

**3.15.17.6 SYSCCTL Registers to Driverlib Functions**
**Table 3-232. SYSCCTL Registers to Driverlib Functions**

File	Driverlib Function
<b>PARTIDL</b>	
sysctl.c	SysCtl_getDeviceParametric
<b>PARTIDH</b>	
sysctl.c	SysCtl_getDeviceParametric
<b>REVID</b>	
sysctl.h	SysCtl_getDeviceRevision
<b>TRIMERRSTS</b>	
-	
<b>SOFTPRES2</b>	
-	See SOFTPRES0
<b>SOFTPRES3</b>	
-	See SOFTPRES0
<b>SOFTPRES4</b>	
-	See SOFTPRES0
<b>SOFTPRES7</b>	
-	See SOFTPRES0
<b>SOFTPRES8</b>	
-	See SOFTPRES0
<b>SOFTPRES9</b>	
-	See SOFTPRES0
<b>SOFTPRES10</b>	
-	See SOFTPRES0
<b>SOFTPRES13</b>	
-	See SOFTPRES0
<b>SOFTPRES14</b>	
-	See SOFTPRES0
<b>SOFTPRES21</b>	
-	
<b>SOFTPRES27</b>	
-	
<b>SOFTPRES28</b>	
-	
<b>TAP_STATUS</b>	
-	
<b>ECAPTYPE</b>	
sysctl.c	SysCtl_configureType

**Table 3-232. SYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.c	SysCtl_isConfigTypeLocked
<b>CLKCFGLOCK1</b>	
sysctl.c	SysCtl_lockClkConfig
<b>CLKSRCCTL1</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
sysctl.c	SysCtl_selectXTAL
sysctl.c	SysCtl_selectXTALSingleEnded
sysctl.c	SysCtl_selectOscSource
sysctl.h	SysCtl_setIntOSC2_Mode
sysctl.h	SysCtl_enableWatchdogInHalt
sysctl.h	SysCtl_disableWatchdogInHalt
<b>CLKSRCCTL2</b>	
can.h	CAN_selectClockSource
<b>CLKSRCCTL3</b>	
sysctl.h	SysCtl_selectClockOutSource
<b>SYSPLLCTL1</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
<b>SYSPLLMULT</b>	
sysctl.c	SysCtl_getClock
sysctl.c	SysCtl_setClock
<b>SYSPLLSTS</b>	
sysctl.c	SysCtl_setClock
<b>SYSCLKDIVSEL</b>	
sysctl.c	SysCtl_getClock
sysctl.h	SysCtl_setPLLSysClk
<b>XCLKOUTDIVSEL</b>	
sysctl.h	SysCtl_setXCik
<b>LOSPCP</b>	
sysctl.c	SysCtl_getLowSpeedClock
sysctl.h	SysCtl_setLowSpeedClock
<b>MCDCR</b>	
sysctl.h	SysCtl_enableMCD
sysctl.h	SysCtl_disableMCD
sysctl.h	SysCtl_isMCDClockFailureDetected
sysctl.h	SysCtl_resetMCD
sysctl.h	SysCtl_connectMCDClockSource
sysctl.h	SysCtl_disconnectMCDClockSource
<b>X1CNT</b>	
sysctl.c	SysCtl_pollX1Counter
sysctl.h	SysCtl_getExternalOscCounterValue
sysctl.h	SysCtl_clearExternalOscCounterValue
<b>XTALCR</b>	
sysctl.c	SysCtl_setClock

**Table 3-232. SYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.c	SysCtl_selectXTAL
sysctl.c	SysCtl_selectXTALSingleEnded
sysctl.h	SysCtl_setExternalOscMode
sysctl.h	SysCtl_turnOnOsc
sysctl.h	SysCtl_turnOffOsc
<b>XTALCR2</b>	
sysctl.c	SysCtl_selectXTAL
<b>CLKFAILCFG</b>	
-	
<b>CPUSYSLOCK1</b>	
sysctl.c	SysCtl_lockSysConfig
<b>CPUSYSLOCK2</b>	
-	
<b>PIEVERRADDR</b>	
sysctl.h	SysCtl_getPIEErrAddr
<b>PCLKCR0</b>	
sysctl.h	SysCtl_enablePeripheral
sysctl.h	SysCtl_disablePeripheral
<b>PCLKCR2</b>	
-	See PCLKCR0
<b>PCLKCR3</b>	
-	See PCLKCR0
<b>PCLKCR4</b>	
-	See PCLKCR0
<b>PCLKCR7</b>	
-	See PCLKCR0
<b>PCLKCR8</b>	
-	See PCLKCR0
<b>PCLKCR9</b>	
-	See PCLKCR0
<b>PCLKCR10</b>	
-	See PCLKCR0
<b>PCLKCR13</b>	
-	See PCLKCR0
<b>PCLKCR14</b>	
-	See PCLKCR0
<b>PCLKCR21</b>	
-	
<b>PCLKCR27</b>	
-	
<b>SIMRESET</b>	
sysctl.h	SysCtl_simulateReset
<b>LPMCR</b>	
sysctl.h	SysCtl_enterIdleMode
sysctl.h	SysCtl_enterStandbyMode

**Table 3-232. SYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.h	SysCtl_enterHaltMode
sysctl.h	SysCtl_setStandbyQualificationPeriod
sysctl.h	SysCtl_enableWatchdogStandbyWakeup
sysctl.h	SysCtl_disableWatchdogStandbyWakeup
<b>GPIOLPMSEL0</b>	
sysctl.h	SysCtl_enableLPMWakeupPin
sysctl.h	SysCtl_disableLPMWakeupPin
<b>GPIOLPMSEL1</b>	
sysctl.h	SysCtl_enableLPMWakeupPin
sysctl.h	SysCtl_disableLPMWakeupPin
<b>TMR2CLKCTL</b>	
cputimer.h	CPUTimer_selectClockSource
sysctl.h	SysCtl_setCputimer2Clk
<b>RESCCLR</b>	
sysctl.h	SysCtl_clearResetCause
sysctl.h	SysCtl_clearWatchdogResetStatus
<b>RESC</b>	
sysctl.h	SysCtl_getResetCause
sysctl.h	SysCtl_clearResetCause
sysctl.h	SysCtl_getWatchdogResetStatus
sysctl.h	SysCtl_clearWatchdogResetStatus
<b>USER_REG1_SYSRN</b>	
sysctl.h	SysCtl_setUserRegister
sysctl.h	SysCtl_getUserRegister
<b>USER_REG2_SYSRN</b>	
-	
<b>USER_REG1_XRSN</b>	
-	
<b>USER_REG2_XRSN</b>	
-	
<b>USER_REG1_PORESETN</b>	
-	
<b>USER_REG2_PORESETN</b>	
-	
<b>USER_REG3_PORESETN</b>	
-	
<b>USER_REG4_PORESETN</b>	
-	
<b>SCSR</b>	
sysctl.h	SysCtl_setWatchdogMode
sysctl.h	SysCtl_isWatchdogInterruptActive
sysctl.h	SysCtl_clearWatchdogOverride
<b>WDCNTR</b>	
sysctl.h	SysCtl_getWatchdogCounterValue
<b>WDKEY</b>	

**Table 3-232. SYSCTL Registers to Driverlib Functions (continued)**

File	Driverlib Function
sysctl.h	SysCtl_serviceWatchdog
sysctl.h	SysCtl_enableWatchdogReset
sysctl.h	SysCtl_resetWatchdog
<b>WDCR</b>	
sysctl.h	SysCtl_resetDevice
sysctl.h	SysCtl_disableWatchdog
sysctl.h	SysCtl_enableWatchdog
sysctl.h	SysCtl_isWatchdogEnabled
sysctl.h	SysCtl_setWatchdogPredivider
sysctl.h	SysCtl_setWatchdogPrescaler
<b>WDWCR</b>	
sysctl.h	SysCtl_setWatchdogWindowValue
<b>SYNCSELECT</b>	
sysctl.h	SysCtl_setSyncOutputConfig
<b>ADCSOCOUTSELECT</b>	
sysctl.h	SysCtl_enableExtADCSOCSource
sysctl.h	SysCtl_disableExtADCSOCSource
<b>SYNCSOCLOCK</b>	
sysctl.h	SysCtl_lockExtADCSOCSelect
sysctl.h	SysCtl_lockSyncSelect
<b>SYS_ERR_INT_FLG</b>	
sysctl.h	SysCtl_getInterruptStatus
<b>SYS_ERR_INT_CLR</b>	
sysctl.h	SysCtl_clearInterruptStatus
<b>SYS_ERR_INT_SET</b>	
sysctl.h	SysCtl_setInterruptStatus
<b>SYS_ERR_MASK</b>	
sysctl.h	SysCtl_getInterruptStatusMask
sysctl.h	SysCtl_setInterruptStatusMask

**3.15.17.7 XINT Registers to Driverlib Functions****Table 3-233. XINT Registers to Driverlib Functions**

File	Driverlib Function
<b>1CR</b>	
gpio.c	GPIO_setInterruptPin
gpio.h	GPIO_setInterruptType
gpio.h	GPIO_getInterruptType
gpio.h	GPIO_enableInterrupt
gpio.h	GPIO_disableInterrupt
gpio.h	GPIO_getInterruptCounter
<b>2CR</b>	
-	See 1CR
<b>3CR</b>	
-	See 1CR
<b>4CR</b>	

**Table 3-233. XINT Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See 1CR
<b>5CR</b>	
-	See 1CR
<b>1CTR</b>	
gpio.h	GPIO_getInterruptCounter
<b>2CTR</b>	
-	
<b>3CTR</b>	
-	

This page intentionally left blank.

Chapter 4  
**ROM Code and Peripheral Booting**

---



This chapter explains the boot procedure, the available boot modes, and the various details of the ROM code including memory maps, initializations, reset handling, and status information.

<b>4.1 Introduction</b> .....	<b>352</b>
<b>4.2 Device Boot Sequence</b> .....	<b>353</b>
<b>4.3 Device Boot Modes</b> .....	<b>353</b>
<b>4.4 Device Boot Configurations</b> .....	<b>354</b>
<b>4.5 Device Boot Flow Diagrams</b> .....	<b>359</b>
<b>4.6 Device Reset and Exception Handling</b> .....	<b>363</b>
<b>4.7 Boot ROM Description</b> .....	<b>364</b>
<b>4.8 Application Notes for Using the Bootloaders</b> .....	<b>390</b>

## 4.1 Introduction

The purpose of this chapter is to explain the boot read-only memory (ROM) code functionality for the CPU core, including the boot procedure. This chapter also discusses the functions and features of the boot ROM code, and provides details about the ROM memory-map contents. On every reset, the device executes a boot sequence in the ROM depending on the reset type and boot configuration. This sequence initializes the device to run the application code. For the CPU, the boot ROM also contains peripheral bootloaders that can be used to load an application into RAM. These bootloaders can be disabled for safety or security purposes.

See [Table 4-1](#) for details on available boot features for the C28x CPU. Additionally, [Table 4-2](#) shows the sizes of the various ROMs on the device.

For details on the security APIs provided, refer to [Section 4.7.9](#).

Various tables are provided in ROM for use in software library, refer to [Section 4.7.6](#) for more details.

**Table 4-1. Boot System Overview**

Boot Feature	CPU
Initial boot process	Device reset
Boot mode selection	GPIOs
Boot modes supported	Flash boot Secure Flash boot RAM boot
Peripheral boot loaders supported	Parallel IO SCI / Wait CAN I2C SPI

**Table 4-2. ROM Memory**

ROM	CPU Size
Secure and Unsecure boot ROM	64KB

### 4.1.1 ROM Related Collateral

#### Foundational Materials

- [Bootloading 101](#) (Video)

#### Getting Started Materials

- [Secure BOOT On C2000 Device Application Report](#)

#### Expert Materials

- [C2000 Software Controlled Firmware Update Process Application Report](#)

## 4.2 Device Boot Sequence

Table 4-3 describes the general boot ROM procedure each time the CPU core is reset.

During boot, boot ROM code updates a boot status location in RAM that details the actions taken during this process. Refer to Section 4.7.11 for more details.

**Table 4-3. Device Boot ROM Sequence**

Step	CPU Action
1	After reset, perform watchdog initialization
2	Clock configuration and Flash power-up
3	Peripheral trimming and device configuration registers are loaded from OTP.
4	On power-on reset (POR), all RAMs are initialized.
5	Non-maskable interrupt (NMI) handling is enabled and DCSM initialization is performed.
6	Device calibration is performed; trimming the specified peripherals with set OTP values.
7	Determine if polling the GPIO pins are needed for determining the boot mode and, if so, read the boot mode GPIO pins to determine the boot mode to run.
8	Based on the boot mode and options, the appropriate boot sequence is executed. Refer to Section 4.5.1 for a flow chart of the boot sequences.

## 4.3 Device Boot Modes

This section explains the default boot modes, as well as all the available custom boot modes supported on this device. The boot ROM uses the boot mode select, general purpose input/output (GPIO) pins to determine the boot mode configuration.

### 4.3.1 Default Boot Modes

Table 4-4 shows the boot mode options available for selection by the default boot mode select pins. Users have the option to program the device to customize the boot modes selectable in the boot-up table as well as the boot mode select pin GPIOs used.

**Table 4-4. Device Default Boot Modes**

Boot Mode	GPIO24 (Default boot mode select pin 1)	GPIO32 (Default boot mode select pin 0)
Parallel IO	0	0
SCI / Wait Boot <sup>(1)</sup>	0	1
CAN	1	0
Flash	1	1

(1) SCI boot mode is used as a wait boot mode as long as SCI continues to wait for an 'A' or 'a' during the SCI autobaud lock process.

Refer to Section 4.7.7.1 for functional details of the boot modes.

Refer to Section 4.7.8 for GPIOs used for selecting the boot modes.

Refer to Section 4.4 for details of boot configurations.

#### Note

All the peripheral boot modes that are supported use the first instance of the peripheral module (SCIA, SPIA, I2CA, CANA, and so forth). Whenever these boot modes are referred to in this chapter, such as SCI boot, the mode is actually referring to the first module instance, which means the SCI boot on the SCIA port. The same applies to the other peripheral boot modes.

### 4.3.2 Custom Boot Modes

Once the user programs a custom boot table in user OTP, an entry in the custom table is used for booting. Users can customize the boot mode select pins in the end system design by programming the BOOTPIN\_CONFIG location in user OTP. This allows customers to use 0, 1, 2, or 3 boot mode select pins as needed. You can also customize the boot definition table and indicate which location to boot from by programming the boot mode definition table in the BOOTDEF location of user OTP. [Table 4-5](#) shows the options for various boot modes.

**Table 4-5. Custom Boot Modes**

Boot Mode Number	Boot Modes
0	Parallel
1	SCI / Wait
2	CAN
3	Flash
4	Wait
5	RAM
6	SPI
7	I2C
10	Secure Flash

## 4.4 Device Boot Configurations

This section details what boot configurations are available and how to configure them. This device supports from zero boot mode select pins up to three boot mode select pins and from one configured boot mode up to eight configured boot modes.

To change and configure the device from the default settings to custom settings for your application, use the following process:

1. Determine all the various ways you want application to be able to boot. (For example: Primary boot option of Flash boot for your main application, secondary boot option of CAN boot for firmware updates, tertiary boot option of SCI boot for debugging, etc)
2. Based on the number of boot modes needed, determine how many boot mode select pins (BMSPs) are required to select between your selected boot modes. (For example: Two BMSPs are required to select between three boot mode options)
3. Assign the required BMSPs to a physical GPIO pin. (For example, BMSP0 to GPIO10, BMSP1 to GPIO51, and BMSP2 left as default which is disabled). Refer to [Section 4.4.1](#) for all the details on performing these configurations.
4. Assign the determined boot mode definitions to indexes in your custom boot table that correlate to the decoded value of the BMSPs. For example, BOOTDEF0=Boot to Flash, BOOTDEF1=CAN Boot, BOOTDEF2=SCI Boot; all other BOOTDEFx are left as default/nothing). Refer to [Section 4.4.2](#) for all the details on setting up and configuring the custom boot mode table.

Additionally, [Section 4.4.3](#) provides some example use cases on how to configure the BMSPs and custom boot tables.

#### 4.4.1 Configuring Boot Mode Pins

This section explains how the boot mode select pins are customized by the user, by programming the BOOTPIN-CONFIG location (refer to [Table 4-6](#)), in the user-configurable dual-zone security module (DCSM) OTP. The location in the DCSM OTP is Z1-OTP-BOOTPIN-CONFIG or Z2-OTP-BOOTPIN-CONFIG. When debugging, EMU-BOOTPIN-CONFIG is the emulation equivalent of Z1-OTP-BOOTPIN-CONFIG/Z2-OTP-BOOTPIN-CONFIG, and can be programmed to experiment with different boot modes without writing to OTP. The device can be programmed to use **zero**, **one**, **two**, or **three** boot mode select pins as needed.

---

#### Note

When using Z2-OTP-BOOTPIN-CONFIG, the configurations programmed in this location take priority over the configurations in Z1-OTP-BOOTPIN-CONFIG. It is recommended to use Z1-OTP-BOOTPIN-CONFIG first and then, if OTP configurations need to be altered, switch to using Z2-OTP-BOOTPIN-CONFIG.

---

**Table 4-6. BOOTPIN-CONFIG Bit Fields**

Bit	Name	Description
31:24	Key	Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid.
23:16	Boot Mode Select Pin 2 (BMSP2)	Refer to BMSP0 description.
15:8	Boot Mode Select Pin 1 (BMSP1)	Refer to BMSP0 description.
7:0	Boot Mode Select Pin 0 (BMSP0)	Set to the GPIO pin to be used during boot (up to 255). 0x0 = GPIO0, 0x01 = GPIO1, and so on. Writing 0xFF disables this BMSP and this pin is no longer used to select the boot mode.

---

#### Note

GPIO 224 to 253 are analog pins, but digital inputs are possible on these pins provided the software writes to the GPIOHAMSEL register bits.

The following GPIOs cannot be used as a BMSP. If selected for a particular BMSP, the boot ROM automatically selects the factory default GPIOs for BMSP0 and BMSP1. Factory default for BMSP2 is 0xFF, which disables the BMSP.

- GPIO 14 and GPIO 15 (Not available on any package)
  - GPIO 25 to GPIO 27 (Not available on any package)
  - GPIO 30, GPIO 31, GPIO 34, and GPIO 38 (Not available on any package)
  - GPIO 42 to GPIO 58 (Not available on any package)
  - GPIO 62 to GPIO 223 (Not available on any package)
-

**Table 4-7. Standalone Boot Mode Select Pin Decoding**

BOOTPIN_CONFIG Key	BMSP0	BMSP1	BMSP2	Realized Boot Mode
!= 0x5A	Don't Care	Don't Care	Don't Care	Boot as defined by the factory default BMSPs.
= 0x5A	0xFF	0xFF	0xFF	Boot as defined in the boot table for boot mode 0 (All BMSPs disabled).
	Valid GPIO	0xFF	0xFF	Boot as defined by the value of BMSP0 (BMSP1 and BMSP2 disabled).
	0xFF	Valid GPIO	0xFF	Boot as defined by the value of BMSP1 (BMSP0 and BMSP2 disabled).
	0xFF	0xFF	Valid GPIO	Boot as defined by the value of BMSP2 (BMSP0 and BMSP1 disabled)
	Valid GPIO	Valid GPIO	0xFF	Boot as defined by the values of BMSP0 and BMSP1 (BMSP2 disabled).
	Valid GPIO	0xFF	Valid GPIO	Boot as defined by the values of BMSP0 and BMSP2 (BMSP1 disabled).
	0xFF	Valid GPIO	Valid GPIO	Boot as defined by the values of BMSP1 and BMSP2 (BMSP0 disabled).
	Valid GPIO	Valid GPIO	Valid GPIO	Boot as defined by the values of BMSP0, BMSP1, and BMSP2.
	Invalid GPIO	Valid GPIO	Valid GPIO	BMSP0 is reset to the factory default BMSP0 GPIO. Boot as defined by the values of BMSP0, BMSP1, and BMSP2.
	Valid GPIO	Invalid GPIO	Valid GPIO	BMSP1 is reset to the factory default BMSP1 GPIO. Boot as defined by the values of BMSP0, BMSP1, and BMSP2.
Valid GPIO	Valid GPIO	Invalid GPIO	BMSP2 is reset to the factory default state, which is disabled. Boot as defined by the values of BMSP0 and BMSP1.	

#### Note

When decoding the boot mode, BMSP0 is the least-significant bit and BMSP2 is the most-significant bit of the boot table index value. It is recommended when disabling BMSPs to start with disabling BMSP2. For example, in an instance when only using BMSP2 (BMSP1 and BMSP0 are disabled), then only the boot table indexes of 0 and 4 are selectable. In the instance when using only BMSP0, then the selectable boot table indexes are 0 and 1.

#### 4.4.2 Configuring Boot Mode Table Options

This section explains how to configure the boot definition table, BOOTDEF, for the device and the associated boot options. The 64-bit location is located in user-configurable DCSM OTP in the Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH locations. When debugging, EMU-BOOTDEF-LOW and EMU-BOOTDEF-HIGH are the emulation equivalents of Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH, and can be programmed to experiment with different boot mode options without writing to OTP. The range of customization to the boot definition table depends on how many boot mode select pins (BMSP) are being used. For example, 0 BMSPs equals to 1 table entry, 1 BMSP equals to 2 table entries, 2 BMSPs equals to 4 table entries, and 3 BMSPs equals to 8 table entries. Refer to [Section 4.4.3](#) for examples on how to setup the BOOTPIN\_CONFIG and BOOTDEF values.

#### Note

The locations Z2-OTP-BOOTDEF-LOW and Z2-OTP-BOOTDEF-HIGH are used instead of Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH locations when Z2-OTP-BOOTPIN-CONFIG is configured. Refer to [Section 4.4.1](#) for more details on BOOTPIN\_CONFIG usage.

**Table 4-8. BOOTDEF Bit Fields**

BOOTDEF Name	Byte Position	Name	Description
BOOT_DEF0	7:0	[3:0] BOOT_DEF0 Mode	Set the boot mode number from <a href="#">Section 4.3.2</a> . Any unsupported boot mode causes the device to either go to wait boot (debugger connected) or boot to Flash (standalone).
		[7:4] BOOT_DEF0 Options	Set alternate/additional boot options. This can include changing the GPIOs for a particular boot peripheral or specifying a different Flash entry point. Refer to <a href="#">Section 4.7.8</a> for valid BOOTDEF values to set in the table.
BOOT_DEF1	15:8	BOOT_DEF1 Mode/ Options	Refer to BOOT_DEF0 description.
BOOT_DEF2	23:16	BOOT_DEF2 Mode/ Options	
BOOT_DEF3	31:24	BOOT_DEF3 Mode/ Options	
BOOT_DEF4	39:32	BOOT_DEF4 Mode/ Options	
BOOT_DEF5	47:40	BOOT_DEF5 Mode/ Options	
BOOT_DEF6	55:48	BOOT_DEF6 Mode/ Options	
BOOT_DEF7	63:56	BOOT_DEF7 Mode/ Options	

### 4.4.3 Boot Mode Example Use Cases

This section demonstrates some use cases for configuring the boot mode select pins and boot modes.

#### 4.4.3.1 Zero Boot Mode Select Pins

This use case demonstrates a scenario for an application that does not use any boot mode select pins and always has the device boot to Flash.

1. Program the BOOTPIN\_CONFIG location in OTP as follows:
  - Set BOOTPIN\_CONFIG.BMSP0 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP1 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP2 to 0xFF
  - Set BOOTPIN\_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid and use the custom boot table.
2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table. Refer to [Section 4.7.8](#) for valid BOOTDEF values to set in the table.
  - Set BOOTDEF.BOOTDEF0 to 0x03 for booting to Flash (entry address option 0). This sets Flash boot to boot table index 0.
  - Refer to [Section 4.7.2](#) for the available Flash entry points.

**Table 4-9. Zero Boot Pin Boot Table Result**

Boot Mode Table Number	Boot Mode
0	Flash Boot (0x03)

#### 4.4.3.2 One Boot Mode Select Pin

This use case demonstrates a scenario for an application using one boot mode select pin to select between booting to Flash or using CAN boot.

1. Program the BOOTPIN\_CONFIG location in OTP as follows:
  - Set BOOTPIN\_CONFIG.BMSP0 to a user specified GPIO, such as 0x0 for GPIO0
  - Set BOOTPIN\_CONFIG.BMSP1 to 0xFF
  - Set BOOTPIN\_CONFIG.BMSP2 to 0xFF
  - Set BOOTPIN\_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid and use the custom boot table.
2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table. Refer to [Section 4.7.8](#) for valid BOOTDEF values to set in the table.
  - Set BOOTDEF.BOOTDEF0 to 0x02 for CAN booting. This sets CAN boot to boot table index 0.
  - Set BOOTDEF.BOOTDEF1 to 0x03 for booting to Flash (entry address option 0). This sets Flash boot to boot table index 1.

**Table 4-10. One Boot Pin Boot Table Result**

Boot Mode Table Number	Boot Mode
0	CAN Boot (0x02)
1	Flash Boot (0x03)

#### 4.4.3.3 Three Boot Mode Select Pins

This use case demonstrates a scenario for an application using three boot mode select pins to select between various boot modes in the custom boot table.

1. Program the BOOTPIN\_CONFIG location in OTP as follows:
  - Set BOOTPIN\_CONFIG.BMSP0 to a user specified GPIO, such as 0x0 for GPIO0
  - Set BOOTPIN\_CONFIG.BMSP1 to a user specified GPIO, such as 0x1 for GPIO1
  - Set BOOTPIN\_CONFIG.BMSP2 to a user specified GPIO, such as 0x2 for GPIO2
  - Set BOOTPIN\_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid and use the custom boot table.
2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table. Refer to [Section 4.7.8](#) for valid BOOTDEF values to set in the table.
  - Set BOOTDEF.BOOTDEF0 to 0x02 for CAN booting. This sets CAN boot to boot table index 0.
  - Set BOOTDEF.BOOTDEF1 to 0x03 for booting to Flash (entry address option 0). This sets Flash boot to boot table index 1.
  - Set BOOTDEF.BOOTDEF2 to 0x24 for booting to wait boot (alternate option). This sets wait boot to boot table index 2.
  - Set BOOTDEF.BOOTDEF3 to 0x66 for SPI booting (alternate GPIO option 3). This sets SPI boot to boot table index 3.
  - Set BOOTDEF.BOOTDEF4 to 0x43 for booting to Flash (entry address option 2). This sets Flash boot to boot table index 4.

**Table 4-11. Three Boot Pins Boot Table Result**

Boot Mode Table Number	Boot Mode
0	CAN Boot (0x02)
1	Flash Boot (0x03)
2	Wait Boot - Alt (0x24)
3	SPI - Alt3 (0x66)
4	Flash Boot - Alt2 (0x43)
5, 6, 7	Not used in this example

## 4.5 Device Boot Flow Diagrams

This section details the boot flow diagrams for standalone and emulation boot flows.

### 4.5.1 Boot Flow

Upon reset, the CPU follows the boot flow shown in [Figure 4-1](#). Depending on whether a JTAG debugger is connected to the device, the CPU either continues booting following the emulation boot flow or the standalone boot flow.

---

#### Note

BOR follows same flow as POR.

---

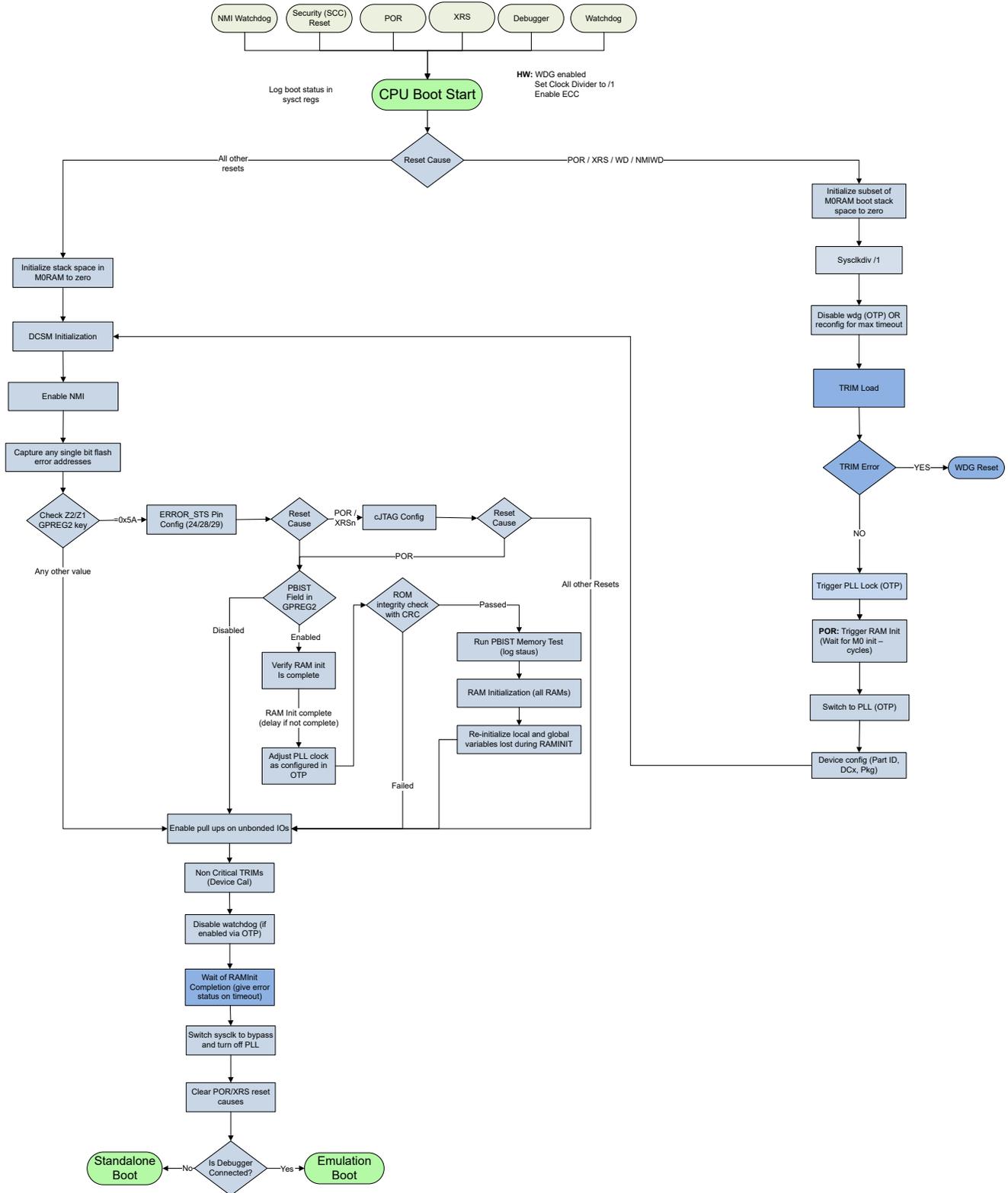


Figure 4-1. Device Boot Flow

### 4.5.2 Emulation Boot Flow

Figure 4-2 shows the emulation boot flow when JTAG debugger is connected.

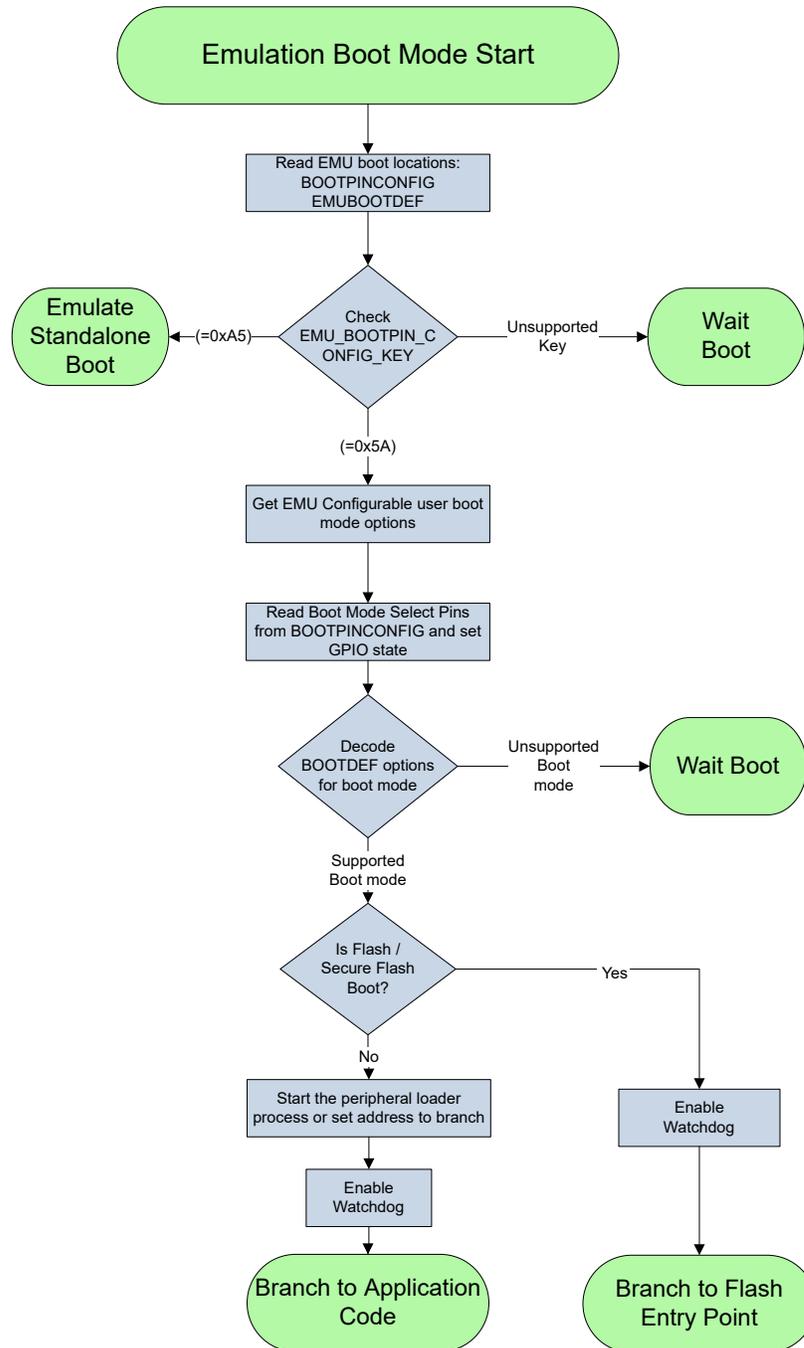


Figure 4-2. Emulation Boot Flow

### 4.5.3 Standalone Boot Flow

Figure 4-3 shows the standalone boot flow when no JTAG debugger is connected to the device.

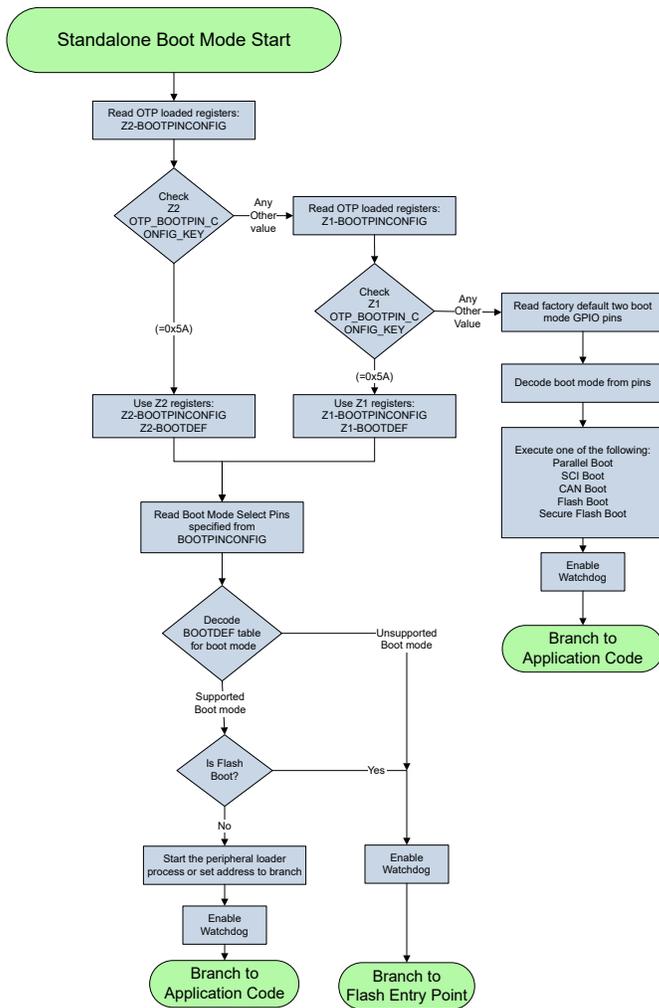


Figure 4-3. CPU Standalone Boot Flow

## 4.6 Device Reset and Exception Handling

### 4.6.1 Reset Causes and Handling

Table 4-12 explains the actions each boot ROM performs upon reset for a specific reset cause.

**Table 4-12. Boot ROM Reset Causes and Actions**

Reset Source	Boot ROM Action
Power on Reset (POR)	<ol style="list-style-type: none"> <li>1. Configure Clock Divider</li> <li>2. Flash Power Up</li> <li>3. Device configuration and trimming</li> <li>4. RAM Initialization</li> <li>5. Continue default boot flow</li> </ol>
External Reset (XRS) Includes: <ul style="list-style-type: none"> <li>• Watchdog Reset</li> <li>• NMI Watchdog Reset</li> <li>• SIMRESET XRS</li> </ul>	<ol style="list-style-type: none"> <li>1. Configure Clock Divider</li> <li>2. Flash Power Up</li> <li>3. Device configuration and trimming</li> <li>4. Clear RAM for boot stack</li> <li>5. Continue default boot flow</li> </ol>
Secure Copy Code (SCC) Reset	<ol style="list-style-type: none"> <li>1. Clear RAM for boot stack</li> <li>2. Continue default boot flow</li> </ol>
SIMRESET	<ol style="list-style-type: none"> <li>1. Clear RAM for boot stack</li> <li>2. Continue default boot flow</li> </ol>
Debugger Reset	<ol style="list-style-type: none"> <li>1. Clear RAM for boot stack</li> <li>2. Continue default boot flow</li> </ol>

### 4.6.2 Exceptions and Interrupts Handling

Table 4-13 explains the actions that the boot ROM performs, if any exceptions occur during boot. The exception handling philosophy in most cases is to log the error and continue booting to reach the application.

**Table 4-13. Boot ROM Exceptions and Actions**

Exception Event Source	Boot ROM Action	Event Logged
Clock Fail	Clear the NMI flag and continue to boot	Yes
RAM Uncorrectable Error ROM Parity Error	Perform RAM initialization and reset the device	Yes <sup>(1)</sup>
Flash Uncorrectable Error	Reset the device	Yes
Embedded Real-time Analysis and Diagnostic (ERAD) NMI	Clear the NMI flag and continue to boot	Yes
RL NMI (CLB)	Clear the NMI flag and continue to boot	Yes
Software NMI error (Software self test error)	Reset the device	No
ITRAP Exception	Record memory address of where the illegal instruction was executed and let device reset	Yes
Unsupported PIE Interrupts	Ignore and continue to boot	No

- (1) A RAM uncorrectable error or ROM parity error clears the boot status information stored in RAM because a RAM initialization is performed to attempt to correct the error. Since the boot status information is erased, this exception can be identified in that a NMIWD reset occurred and all the RAMs are erased.

## 4.7 Boot ROM Description

This section explains the details regarding the device boot ROMs.

### 4.7.1 Boot ROM Configuration Registers

The boot ROM code involves several memory addresses and registers used during execution. There are two sets of configurations; one for emulation and one for standalone boot flow. The emulation locations located in RAM emulate the OTP configurations and can be written to as many times as needed. The user configurable DCSM OTP locations used in the standalone boot flow program the device OTP and hence can only be written once. [Table 4-14](#) details these locations. For bit field configuration details for BOOTPIN-CONFIG and BOOTDEF, see [Section 4.4.1](#) and [Section 4.4.2](#).

Additionally, the boot ROM supports boot configurations from DCSM zone 1 and zone 2 registers. Zone 2 configurations supercede zone 1 configurations, so it is recommended to use zone 1 configurations and use zone 2 as a secondary option.

**Table 4-14. Boot ROM Registers**

Boot Flow	Register Name	Boot ROM Name	Register Address	User OTP Address
Emulation	-	EMU-BOOTPIN-CONFIG	0x0000 0D00	-
	-	EMU-GPREG2	0x0000 0D02	-
	-	EMU-BOOTDEF-LOW	0x0000 0D04	-
	-	EMU-BOOTDEF-HIGH	0x0000 0D06	-
Standalone (Using Z1)	Z1-GPREG1	Z1-OTP-BOOTPIN-CONFIG	0x0005 F008	0x0007 8008
	Z1-GPREG2	Z1-OTP-BOOT-GPREG2	0x0005 F00A	0x0007 800A
	Z1-GPREG3	Z1-OTP-BOOTDEF-LOW	0x0005 F00C	0x0007 800C
	Z1-GPREG4	Z1-OTP-BOOTDEF-HIGH	0x0005 F00E	0x0007 800E
Standalone (Using Z2)	Z2-GPREG1	Z2-OTP-BOOTPIN-CONFIG	0x0005 F088	0x0007 8208
	Z2-GPREG2	Z2-OTP-BOOT-GPREG2	0x0005 F08A	0x0007 820A
	Z2-GPREG3	Z2-OTP-BOOTDEF-LOW	0x0005 F08C	0x0007 820C
	Z2-GPREG4	Z2-OTP-BOOTDEF-HIGH	0x0005 F08E	0x0007 820E

#### 4.7.1.1 GPREG2 Usage and MPOST Configuration

Table 4-15 explains how the bit field values from the user configurable DCSM OTP location, Z1-OTP-BOOT-GPREG2, are decoded by boot ROM.

**Table 4-15. DCSM Zone GPREG2 Bit Fields**

Bit	Name	Description	Boot ROM Action
31:24	Key	Write 0x5A to indicate to the boot ROM code that the bits in this register are valid.	If user sets to 0x5A, boot ROM uses the values in this register. If set to any other value, boot ROM ignores values in this register.
23:8	Reserved	Reserved	No Action
7:6	MPOST <sup>(1)</sup>	0x0 = Run MPOST with PLL disabled (10MHz internal oscillator) 0x1 = Run MPOST with PLL enabled for 95MHz 0x2 = Run MPOST with PLL enabled for 47.5MHz 0x3 = Disable MPOST	When configured to a valid value, MPOST POR memory self-test runs on all device memories
5:4	ERROR_STS_PIN config	0x0 – GPIO24, MUX Option 13 0x1 – GPIO28, MUX Option 13 0x2 – GPIO29, MUX Option 13 0x3 – ERROR_STS function disabled (default)	This indicates which GPIO pin is supposed to be used as ERROR_PIN and boot ROM configures the mux as such for the said pin. The ERROR_STS pin mux configuration is locked by the boot ROM, but not committed.
3:0	CJTAGNODEID	CJTAGNODEID[3:0]	Boot ROM takes this values and programs the lower 4 bits of the CJTAGNODEID register.

- (1) If MPOST is configured to run with PLL enabled and the PLL fails to lock, then the MPOST run is skipped altogether. This does not apply, if MPOST is configured to use INTOSC2 with PLL disabled.

**Note**

Z1-GPREG2 shares ECC with Z1-GPREG1, so users can program both these locations at the same time in User OTP.

#### 4.7.2 Entry Points

This sections gives details about the entry point addresses for various boot modes. These entry points direct the boot ROM what address to branch to at the end of booting as per the selected boot mode.

Table 4-16 gives details about the entry point addresses for Flash boot mode.

Table 4-17 gives details about the entry point addresses for RAM boot mode.

Table 4-18 gives details about the entry point addresses for Secure Flash boot mode.

**Table 4-16. Flash Entry Point Addresses**

Option	BOOTDEFx Value	Flash Sector	Address	Packages Supported
0	0x03	CPU Bank 0 Sector 0	0x0008 0000	All
1	0x23	CPU Bank 0 Sector 32	0x0008 8000	All
2	0x43	CPU Bank 0 End of Sector 63	0x0008 FFF0	All
3	0x63	CPU Bank 0 Sector 64	0x0009 0000	All
4	0x83	CPU Bank 0 End of Sector 96	0x0009 8000	All
5	0xA3	CPU Bank 0 End of Sector 127	0x0009 FFF0	All

**Table 4-17. RAM Entry Point Address**

Option	BOOTDEFx Value	RAM Entry Point	Package Supported
0	0x05	0x0000 0000	All

**Table 4-18. Secure Flash Entry Point Addresses**

Option	BOOTDEFx Value	Flash Sector	Address	Packages Supported
0 (default)	0x0A	CPU Bank 0 Sector 0	0x0008 0000	All
1	0x2A	CPU Bank 0 Sector 32	0x0008 8000	All
2	0x4A	CPU Bank 0 End of Sector 63	0x0008 FFF0	All
3	0x6A	CPU Bank 0 Sector 64	0x0009 0000	All
4	0x8A	CPU Bank 0 End of Sector 96	0x0009 8000	All

### 4.7.3 Wait Points

The wait mode puts the CPU in a loop in the boot ROM code and does not branch to the user application code. The device can enter the wait boot mode either through manually being set or because of some issue during boot up. Using the wait boot mode is recommended when using a debugger to avoid any JTAG issues. There is an ESTOP provided for debugging during Wait boot.

**Table 4-19. Wait Boot Options**

Option	BOOTDEFx Value	Watchdog Status	Package Supported
0 (default)	0x04	Enabled	All
1	0x24	Disabled	All

During boot ROM execution, there are situations where the CPU can enter a wait loop in the code. This state can occur for a variety of reasons. [Table 4-20](#) details the address ranges that the CPU PC register value falls between, if the CPU has entered one of these instances.

Following are the actions for entering wait boot mode:

- Wait boot is set by the user as the boot mode.
- Boot mode is unrecognizable and a debugger is connected to the device.
- The emulation BOOTPIN\_CONFIG key isn't equal to 0xA5 or 0x5A.
- An error occurs during emulation boot and the boot mode pins are decoded with a value not recognized as a valid boot mode.

**Table 4-20. Wait Point Addresses**

Address Range	Description
0x003F D4BC – 0x003F D4C2	In Wait Boot Mode
0x003F E047 – 0x003F E04F	In SCI Boot waiting on autobaud lock
0x003F FE4C – 0x003F FED8	In NMI Handler
0x003F FED9 – 0x003F FF0D	In ITRAP ISR
0x003F D611 - 0x003F D614 0x003F D4FB - 0x003F D512	In Parallel boot waiting for control signal

#### 4.7.4 Secure Flash Boot

Secure Flash boot mode is similar to Flash boot mode in that the boot flow branches to the configured memory address in Flash but only after the Flash memory contents have been authenticated. The Flash authentication uses a Cipher-based Message Authentication Protocol (CMAC) to authenticate 16KB of Flash starting from the configured Flash entry point address. The CMAC calculation requires a user-defined 128-bit key programmed in the CPU User OTP Zone 1 Header OTP CMACKEY bit field. Additionally, you must calculate the golden CMAC tag based on the 16KB Flash memory range and store the tag along with the user code at a hardcoded address in Flash. During secure Flash boot, the calculated CMAC tag is compared to the user golden CMAC tag in Flash to determine the pass/fail status of the CMAC authentication. When authentication passes, boot flow continues and branches to Flash to begin executing the application. When authentication fails, the device is reset.

For the available secure Flash boot entry address options, refer to [Section 4.7.2](#).

For generating the secure Flash golden CMAC tag for CPU, refer to the [TMS320C28x Assembly Language Tools User's Guide](#) within section "Using Secure Flash Boot on TMS320F2838x Devices" for instructions.

---

#### Note

Both the CMAC golden signature and CMAC key are stored in the most significant double format, but each 32-bit section is in little-endian format.

---

Key: 2B7E1516 28AED2A6 ABF71588 09CF4F3C

(MSB is 2B and LSB is 3C)

CMACKEY0 = 0x2B7E1516

CMACKEY1 = 0x28AED2A6

CMACKEY2 = 0xABF71588

CMACKEY3 = 0x09CF4F3C

---

#### Note

You must make sure that the Flash sector that encompasses the configured Flash entry point and the first 16KB of Flash is assigned to Zone 1 for the core setup for secure Flash boot.

Recommended to use device JTAGLOCK when using secure Flash boot.

---

**Table 4-21. Secure Flash Boot Details**

Details	Location Address
CMAC Signature Address	Flash Entry point Address + 0x2
CMAC Key Address (128-bit key)	DCSM Z1 OTP CMACKEY0/1/2/3
Flash Entry Point (Bank 0, Sector 0)	0x0008 0000
Flash Entry Point (Bank 0, Sector 32)	0x0008 8000
Flash Entry Point (Bank 0, End of Sector 63)	0x0008 FFF0
Flash Entry Point (Bank 0, Sector 64)	0x0009 0000
Flash Entry Point (Bank 0, Sector 96)	0x0009 8000
Address Range for CMAC Calculation	Start: Flash Entry Point Address End: Flash Entry Point Address + 16KB

**Table 4-22. Secure Flash Tag and Key Details**

Name	Address	Details
CMA Golden Tag (128-bit)	<b>CPU:</b> <i>Flash Entry Point Address + 0x2</i>	Located in Flash, offset from the entry point address, by 2 words (CPU). When CMAC calculations are performed, the golden tag location in memory is considered all 0xFs. Refer to <a href="#">Example 4-1</a> for an example regarding linker configuration on CPU. Lower memory contains the tag's MSW and higher memory contains the LSW.  <b>Example (on CPU):</b> Tag = 0x00112233 44556677 8899AABB CCDDEEFF Address 0x0 = 0x00112233 Address 0x2 = 0x44556677 Address 0x4 = 0x8899AABB Address 0x6 = 0xCCDDEEFF
CMAC 128-Bit Key	0x0007 8018	Located in CPU Zone 1 User Header OTP (CMACKEY0, CMACKEY1, CMACKEY2, CMACKEY3) CMACKEY0 contains the key's MSW and CMACKEY3 contains the LSW.  <b>Example:</b> Key = 0x00112233 44556677 8899AABB CCDDEEFF CMACKEY0 = 0x00112233 CMACKEY1 = 0x44556677 CMACKEY2 = 0x8899AABB CMACKEY3 = 0xCCDDEEFF

**Table 4-23. Secure Flash Authentication Failure Actions**

CPU	Action on Failed Authentication
C28x CPU	1. Emulation only - Halt debugger (ESTOP) 2. Wait in endless loop (for device reset due to WD reset)

**Table 4-24. Secure Flash on all CPUs Recommended Flow**

Step	Action
1	Secure Flash boot CPU
2	Any Flash beyond the first 16KB from the entry point that is planned for use can be authenticated by you using a different CMAC golden tag embedded at an address somewhere within the already authenticated 16KB of Flash.

APIs for CMAC calculation and authentication is provided as part of ROM. Details are available in [Section 4.7.9](#).

**Example 4-1. Secure Flash CPU1 Linker File Example**

```
MEMORY
{
  /* Code Start branch to _c_int00 */
  BEGIN : origin = 0x80000, length = 0x0002
  /* User calculated golden CMAC tag for Flash Sector 0 */
  GOLDEN_CM_TAG : origin = 0x80002, length = 0x0008
  /* Flash Sector 0 containing application code */
  FLASH_SECTOR_0 : origin = 0x8000A, length = 0x1FF6
  .
  .
  .
}
```

## 4.7.5 Memory Maps

### 4.7.5.1 Boot ROM Memory Maps

Table 4-25 details the ROM memory map including secure and unsecure ROM.

**Table 4-25. Boot ROM Memory Map**

Memory	Origin Address	Length (Words)
ROM Signature	0x003F 9800	0x0002
Version	0x003F 9804	0x0004
IQmath Tables	0x003F 9808	0x1674
FPU32 Fast Tables	0x003F AE7C	0x081A
FPU32 Twiddle Tables	0x003F B696	0x0DF8
SW CRC Table	0x003F C48E	0x0200
Boot Code	0x003F C68E	0x2372
AES Tables	0x003F EA00	0x1400
Interrupt Handlers	0x003F FE00	0x010E
CRC Table	0x003F FFB6	0x0008
Vector Table	0x003F FFBE	0x0042

### 4.7.5.2 Reserved RAM Memory Maps

Table 4-26 details memory usage in RAM that is reserved for boot ROM to use. These memory sections can be reserved in the user application.

**Table 4-26. Reserved RAM Memory Map**

Memory	Description	Origin Address	Length (Words)
RAM	Boot Status, Boot Mode, MPOST Status, Boot Stack	0x0000 0002	0x0126

## 4.7.6 ROM Tables

Table 4-27 details the boot ROM symbol libraries that can be integrated into an application to use the available ROM functions and tables.

**Table 4-27. ROM Symbol Tables**

ROM Symbols	Library Name	Location
ROM Bootloaders and Functions	F280013xCPU_BootROM_Symbols	Under <code>/libraries/boot_rom</code> in <a href="#">C2000Ware</a>
FPU32 Tables	F280013xCPU_BootROM_Symbols	
IQmath	F280013xCPU_IQMathROM_Symbols	

### 4.7.7 Boot Modes and Loaders

The available boot modes and bootloaders supported on this device are detailed in this section.

#### 4.7.7.1 Boot Modes

This section details the available boot modes that do not involve a peripheral boot loader. [Table 4-28](#) details the available boot modes that do not involve a peripheral boot loader.

**Table 4-28. Boot Mode Availability**

Boot Mode	CPU Support
Flash Boot	C28x CPU
RAM Boot	C28x CPU
Wait Boot	C28x CPU
Secure Flash Boot	C28x CPU

##### 4.7.7.1.1 Flash Boot

Flash boot mode branches to the configured memory address in Flash memory. Refer to [Section 4.7.2](#) for all the available Flash address options.

##### 4.7.7.1.2 RAM Boot

RAM boot mode branches to the configured memory address in RAM. Refer to [Section 4.7.2](#) for all the available RAM address options.

##### 4.7.7.1.3 Wait Boot

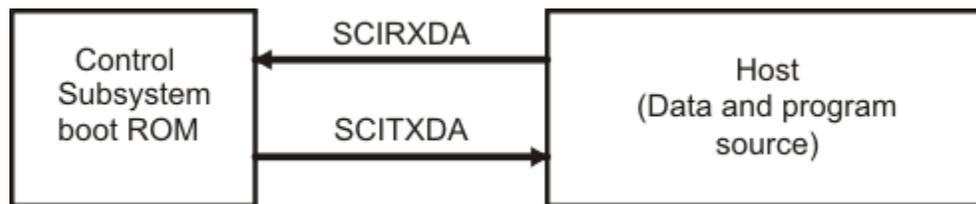
Wait boot mode branches to the memory address as mentioned in [Section 4.7.3](#).

#### 4.7.7.2 Bootloaders

This section details the available boot modes that use a peripheral boot loader. For more specific details on the supported data stream structure used by the following bootloaders, refer to [Section 4.8.1](#).

##### 4.7.7.2.1 SCI Boot Mode

The SCI boot mode asynchronously transfers code from SCI-A to internal memory. This boot mode only supports an incoming 8-bit data stream and follows the data flow as shown in [Figure 4-4](#).



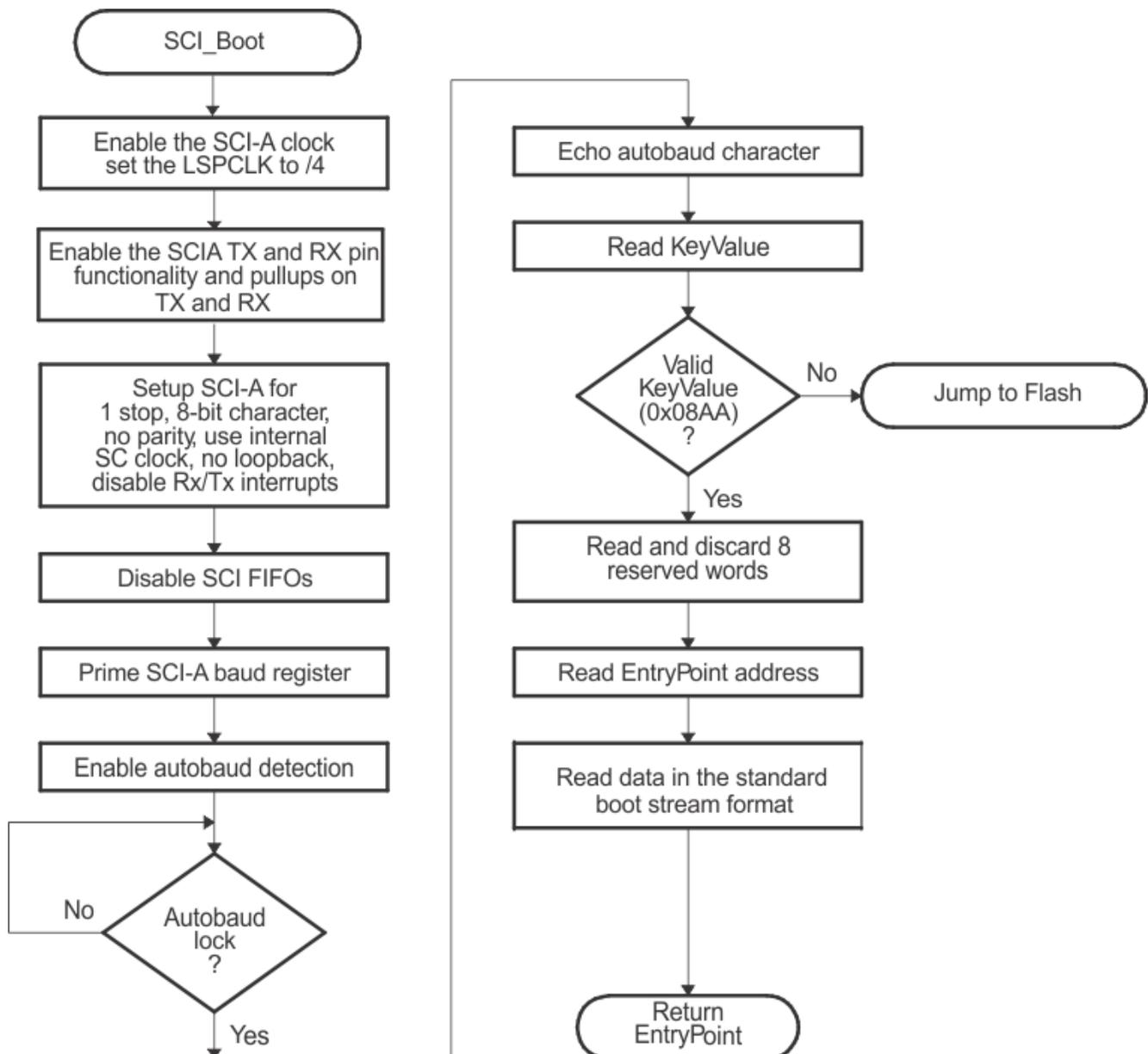
**Figure 4-4. Overview of SCI Bootloader Operation**

The device communicates with the external host by communication through the SCI-A peripheral. The autobaud feature of the SCI port is used to lock baud rates with the host. For this reason the SCI loader is very flexible and you can use a number of different baud rates to communicate with the device.

After each data transfer, the bootloader echoes back the 8-bit character received to the host. This allows the host to check that each character was received by the bootloader.

At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications can work well, this slew rate can limit reliable auto-baud detection at higher baud rates (typically beyond 100kbaud) and cause the auto-baud lock feature to fail. To avoid this, the following is recommended:

1. Achieve a baud-lock between the host and SCI bootloader using a lower baud rate.
2. Load the incoming application or custom loader at this lower baud rate.
3. The host can then handshake with the loaded application to set the SCI baud rate register to the desired high baud rate.



**Figure 4-5. Overview of SCI Boot Function**

4.7.7.2.2 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 shown in Figure 4-6. The SPI bootloader supports an 8-bit data stream and does not support a 16-bit data stream.

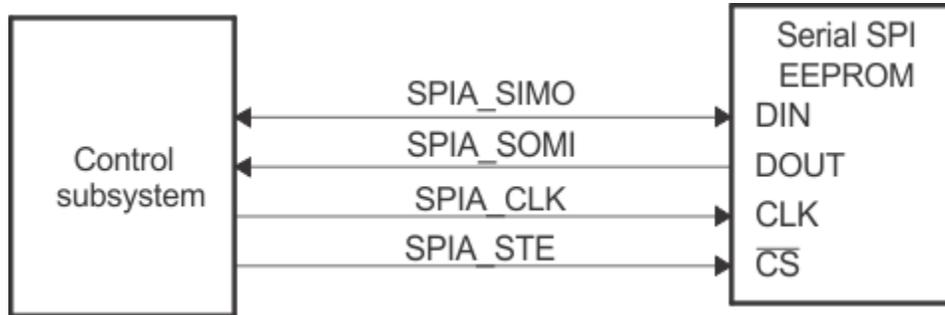


Figure 4-6. Overview of SPI Bootloader Operation

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 set up to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI\_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you could specify a change in baud rate or low speed peripheral clock. Table 4-29 shows the 8-bit data stream used by the SPI.

Table 4-29. 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
...	...
...	Reserved
...	...
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

**Table 4-29. SPI 8-Bit Data Stream (continued)**

Byte	Contents
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:

1. The SPI-A port is initialized.
2. The GPIO pin, as defined by SPI option configured from [Table 4-37](#), is used as a chip-select for the serial SPI EEPROM or Flash.
3. The SPI-A outputs a read command for the serial SPI EEPROM or Flash.
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.
5. The next word fetched must match the key value for an 8-bit data stream (0x08AA). The least significant byte of this word is the byte read first and the most significant byte is the next byte fetched. This is true of all word transfers on the SPI. If the key value does not match, then the load is aborted and the bootloader jumps to Flash.
6. The next two 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 seven words are reserved for future enhancements. The SPI bootloader reads these seven words and discards them.
7. The next two words makeup the 32-bit entry point address where execution continues after the boot load process is complete. This is typically the entry point for the program being downloaded through the SPI port.
8. Multiple blocks of code and data are then copied into memory from the external serial SPI EEPROM through the SPI port. The blocks of code are organized in the standard data stream structure presented earlier. This is done until a block size of 0x0000 is encountered. At that point in time the entry point address is returned to the calling routine that then exits the bootloader and resumes execution at the address specified.

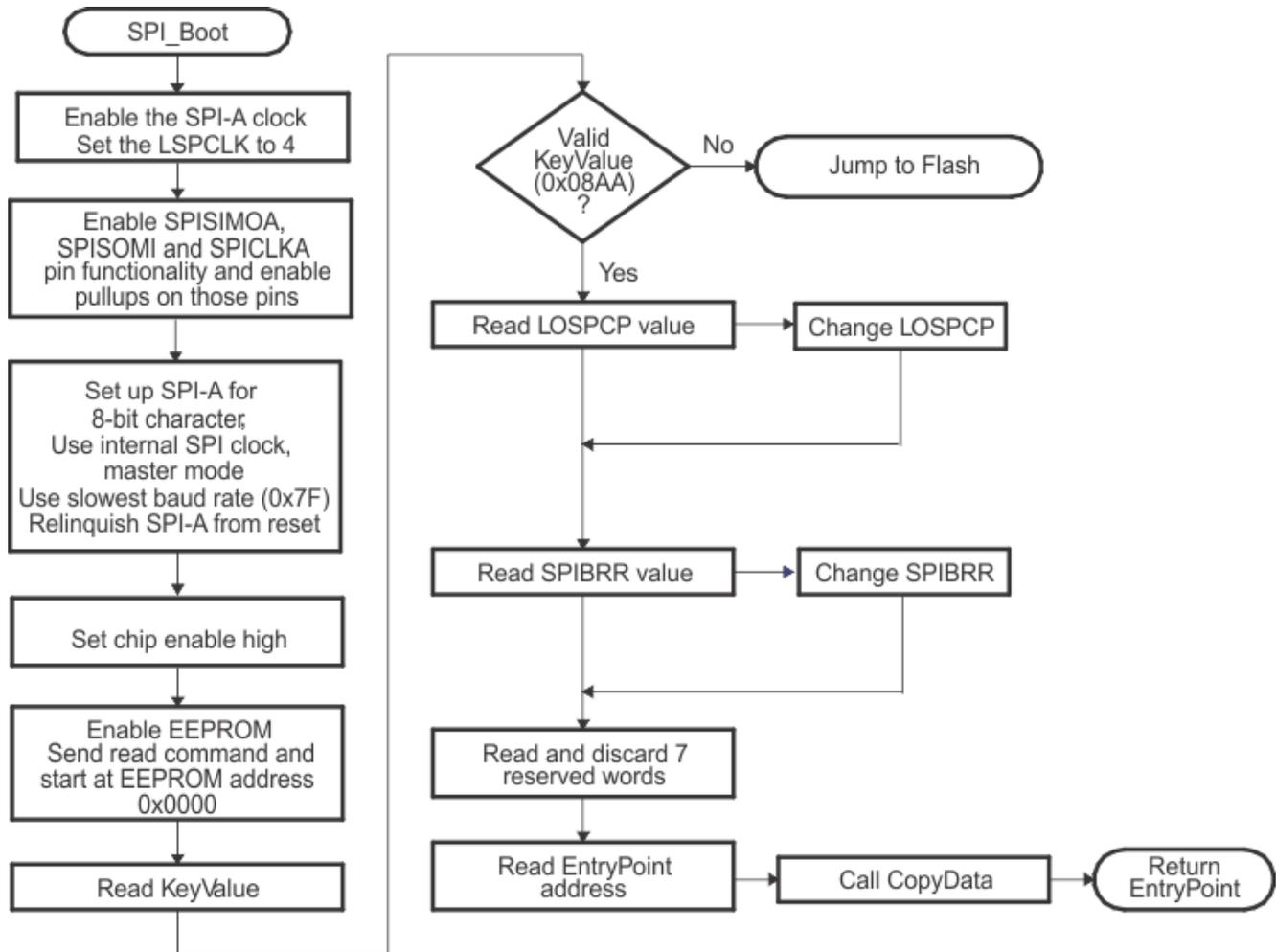
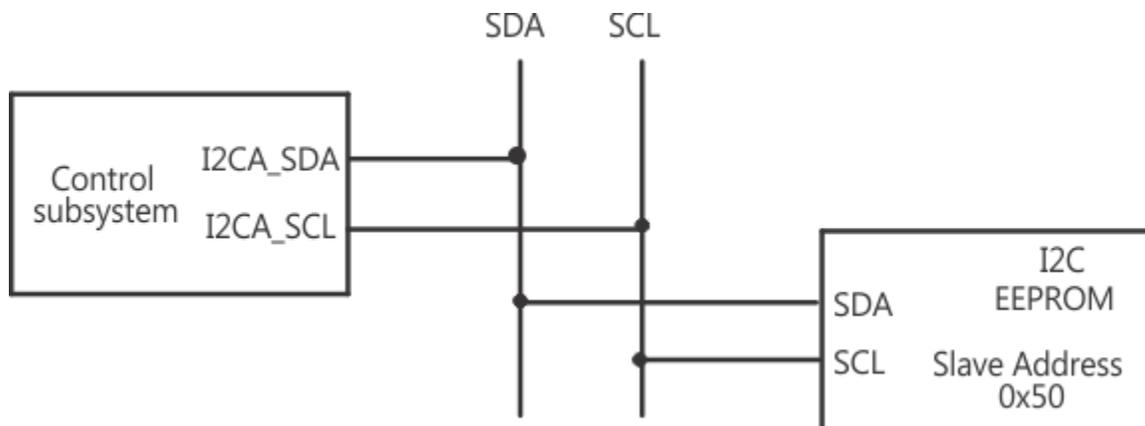


Figure 4-7. Data Transfer from EEPROM Flow

#### 4.7.7.2.3 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 shown in [Figure 4-8](#). The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.



**Figure 4-8. EEPROM Device at Address 0x50**

If the download is to be performed from a device other than an EEPROM, then that device must be set up to operate in the slave mode and mimic the I2C EEPROM. Immediately after entering the I2C boot function, the GPIO pins are configured for I2C-A operation and the I2C is initialized. The following requirements must be met when booting from the I2C module:

- The input frequency to the device must be in the appropriate range.
- The EEPROM must be at slave address 0x50.

The bit-period prescalers (I2CCLKH and I2CCLKL) are configured by the bootloader to run the I2C at a 50 percent duty cycle at 100kHz bit rate (standard I2C mode) when the system clock is 10MHz. These registers can be modified after receiving the first few bytes from the EEPROM. This allows the communication to be increased up to a 400kHz 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 the software is past the bootloader portion of initialization.

The non-acknowledgment bit is checked only during the first message sent to initialize the EEPROM base address. This is to make sure that an EEPROM is present at address 0x50 before continuing. If an EEPROM is not present, the non-acknowledgment bit is not checked during the address phase of the data read messages (I2C\_Get Word). If a non-acknowledgment is received during the data read messages, the I2C bus hangs. [Table 4-30](#) shows the 8-bit data stream used by the I2C.

The I2C EEPROM protocol required by the I2C bootloader is shown in [Figure 4-10](#) and [Figure 4-11](#). The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA), is shown in [Figure 4-10](#). All subsequent reads are shown in [Figure 4-11](#) and are read two bytes at a time.

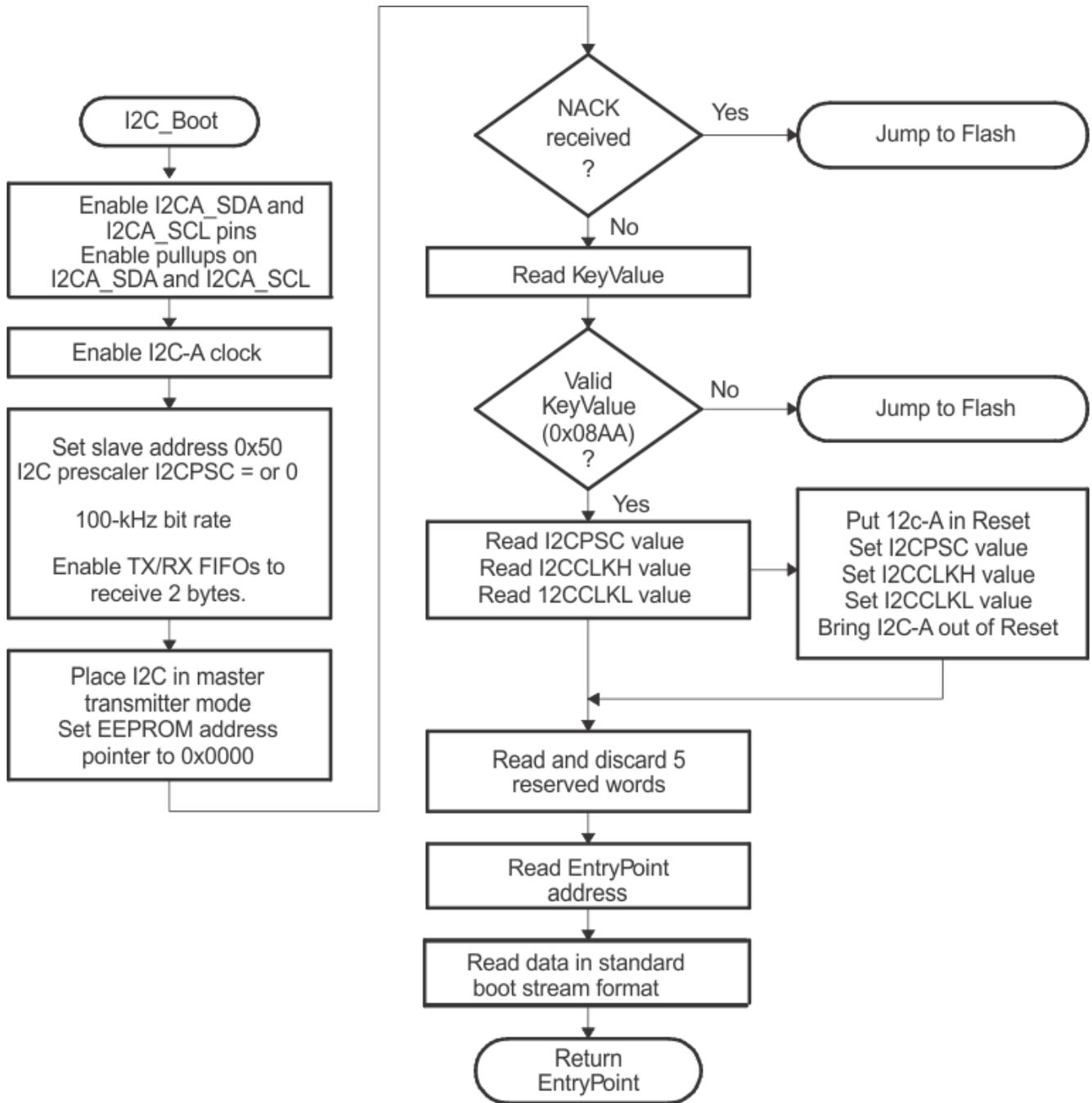
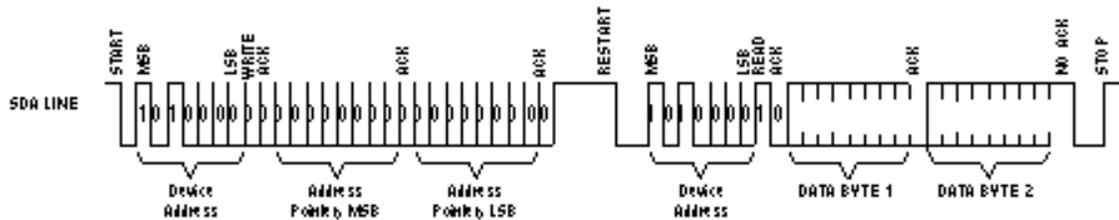
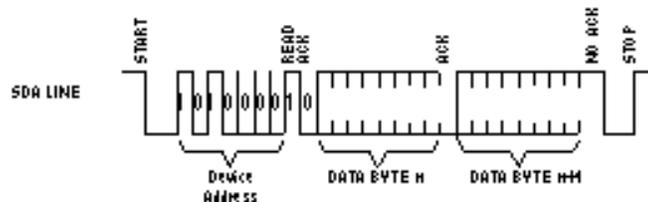


Figure 4-9. Overview of I2C Boot Function

**Table 4-30. 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


**Figure 4-10. Random Read**

**Figure 4-11. Sequential Read**

#### 4.7.7.2.4 Parallel Boot Mode

The parallel general-purpose I/O (GPIO) boot mode asynchronously transfers code from host to C28x internal memory. Each value is 8-bits long and follows the same data flow as outlined in Figure 4-12.

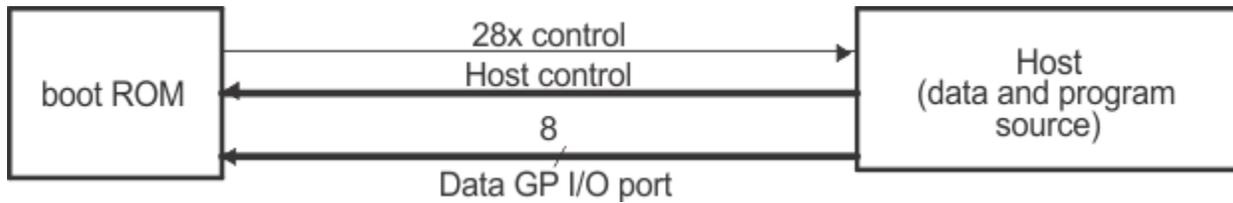


Figure 4-12. Overview of Parallel GPIO Bootloader Operation

The control subsystem communicates with the external host device by polling/driving the Host Control and 28x control lines. The handshake protocol shown in Figure 4-13 must be used to successfully transfer each word using GPIO [D0:D7]. 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 least-significant byte (LSB) is read first followed by the most-significant byte (MSB). In this case, data is read from GPIO[D0:D7].

The 8-bit data stream is shown in Table 4-31.

Table 4-31. Parallel GPIO Boot 8-Bit Data Stream

Bytes	GPIO[D0:D7] (Byte 1 of 2)	GPIO[D0:D7] (Byte 2 of 2)	Description
1 2	AA	08	0x08AA (KeyValue for memory width = 16 bits)
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 the device is ready to begin data transfer by pulling the 28x control pin low. The host load then initiates the data transfer by pulling the DSP control pin low. The complete protocol is shown in Figure 4-13.

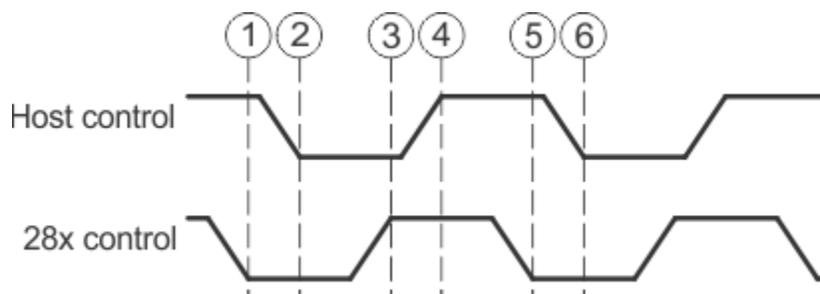


Figure 4-13. Parallel GPIO Bootloader Handshake Protocol

1. The device indicates the device is ready to start receiving data by pulling the 28x control pin low.
2. The bootloader waits until the host puts data on GPIO [D0:D7]. The host signals to the device that data is ready by pulling the host control pin low.
3. The device reads the data and signals the host that the read is complete by pulling the 28x control pin high.
4. The bootloader waits until the host acknowledges the device by pulling the host control pin high.
5. The device again indicates the device is ready for more data by pulling the 28x control pin low.

This process is repeated for each data value to be sent.

Figure 4-14 shows an overview of the Parallel GPIO bootloader flow.

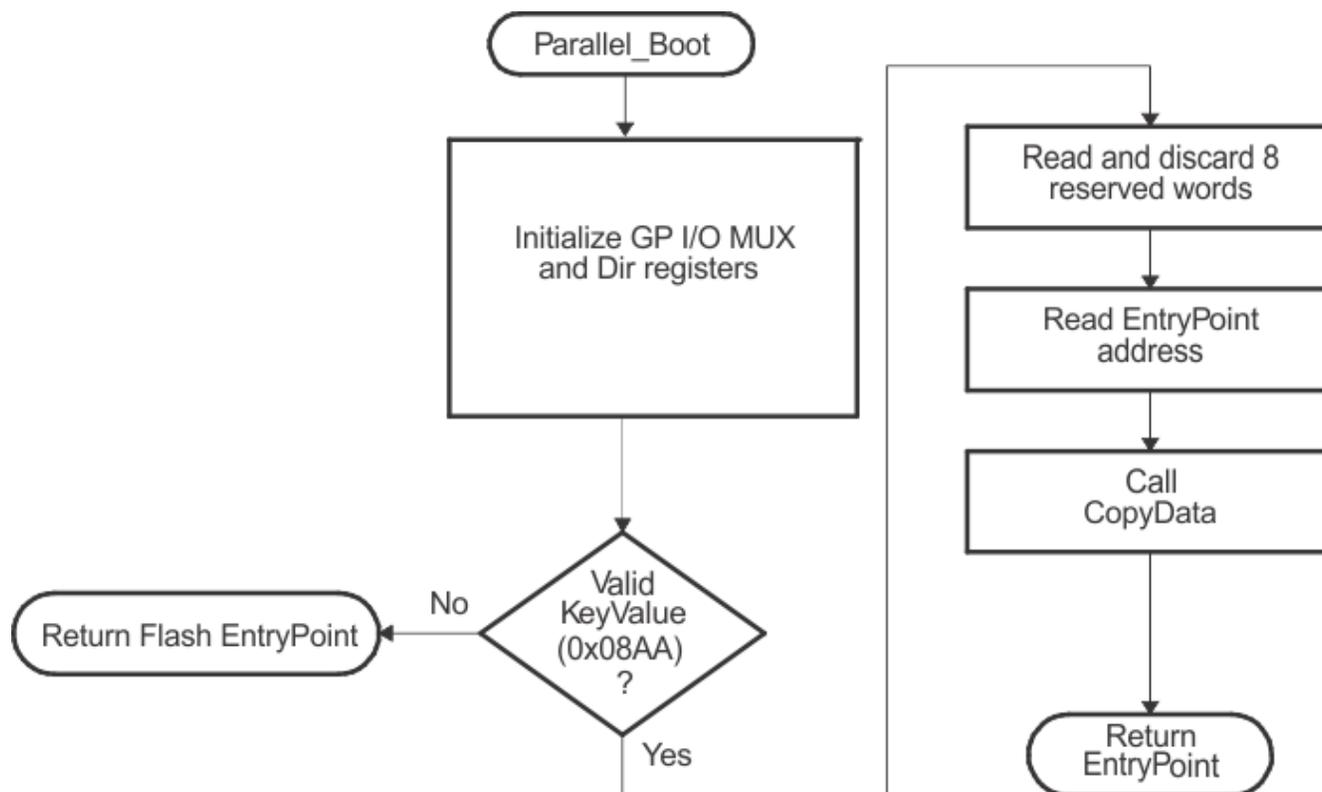


Figure 4-14. Overview of Parallel GPIO Boot Function

Figure 4-15 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host waits for the device and the device in turn waits for the host. In this manner, the protocol works with both a host running faster and a host running slower than the device.

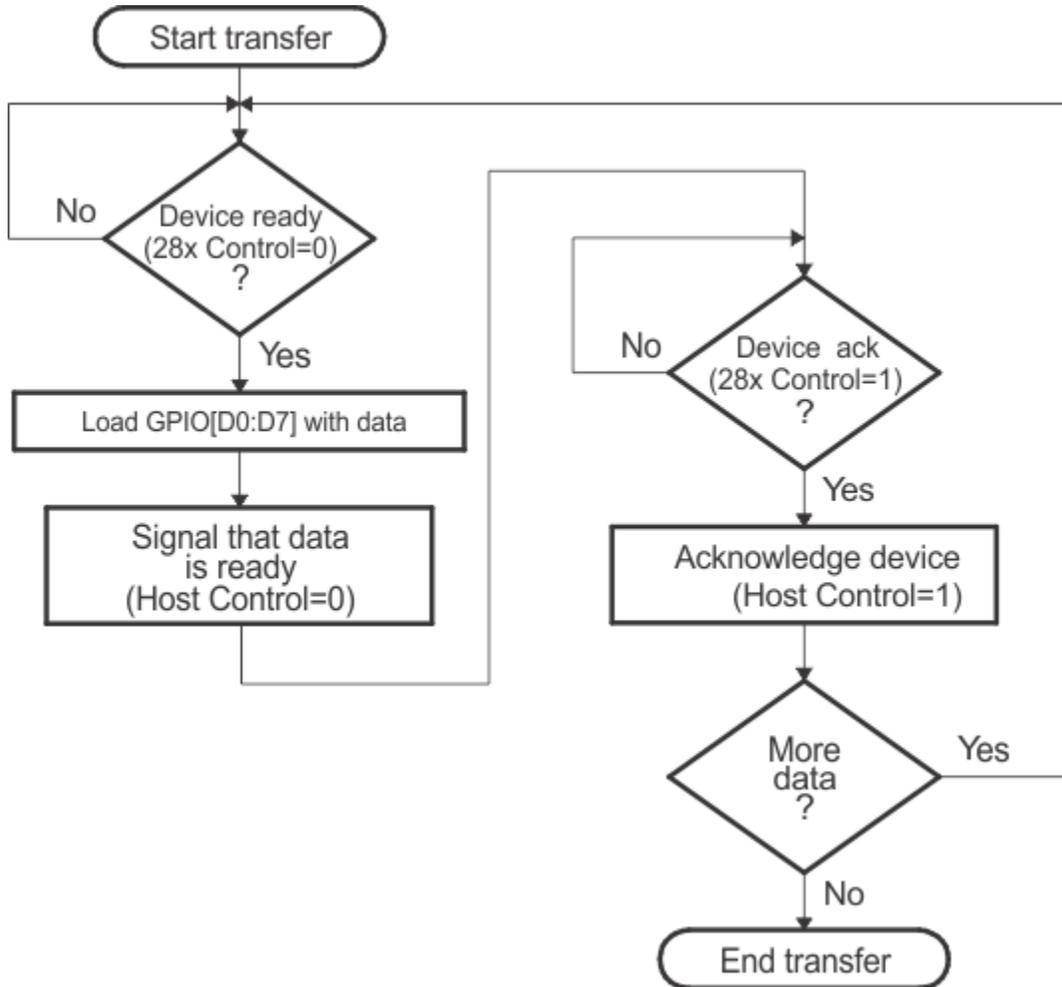


Figure 4-15. Parallel GPIO Mode - Host Transfer Flow

Figure 4-16 shows the flow used to read a single word of data from the parallel port.

- **8-bit data stream**

The 8-bit routine, shown in Figure 4-16, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with D7 in bit position 7 and D6 in bit position 6 as the least-significant byte (LSB) of the word to be fetched. The routine then performs a second read to fetch the most-significant byte (MSB). The routine then performs a second read to fetch the most-significant byte (MSB). The routine then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

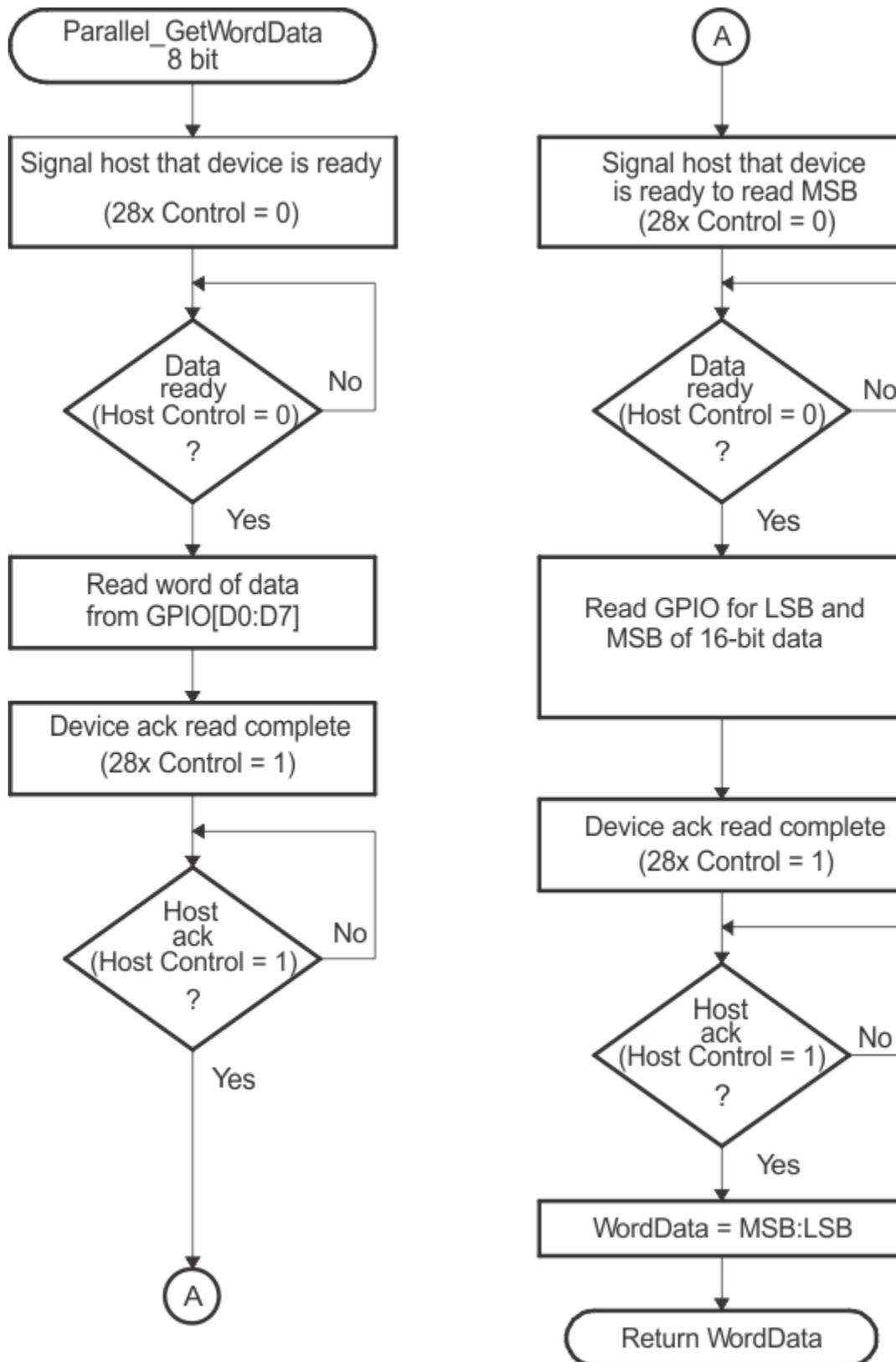
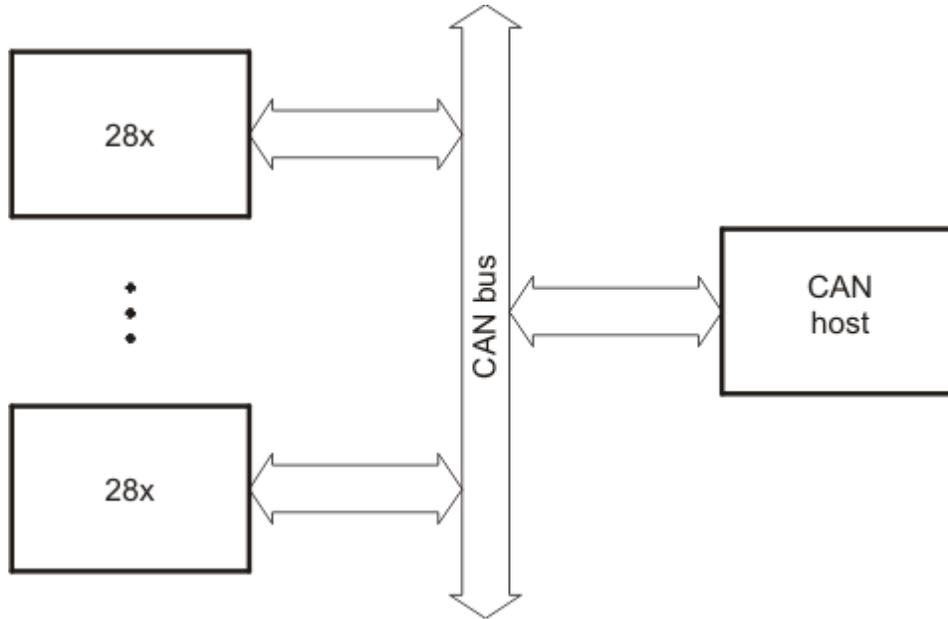


Figure 4-16. 8-Bit Parallel GetWord Function

#### 4.7.7.2.5 CAN Boot Mode

The CAN bootloader asynchronously transfers code from CAN-A to internal memory as shown in [Figure 4-17](#). The host can be any CAN node. The communication is first done with 11-bit standard identifiers (with a MSGID of 0x1) using two bytes per data frame. The host can download a kernel to reconfigure the CAN if higher data throughput is desired.



**Figure 4-17. Overview of CAN-A Bootloader Operation**

The bit timing registers are programmed in such a way that a 100kbps bit rate is achieved with a 20MHz external oscillator, as shown in [Table 4-32](#).

**Table 4-32. Bit-Rate Value for Internal Oscillators**

OSCCLK	SYSCLK	Bit Rate
20MHz	10MHz	100kbps

The SYSCLKOUT values shown are the reset values with the default PLL setting. The BRP and bit-time values are hard-coded to 10 and 20, respectively.

**Note**

The CAN boot loader uses XTAL as the bit clock source and INTOSC2 as the system clock source.

Mailbox 1 is programmed with a standard MSGID of 0x1 for bootloader communication. The CAN host transmits only two bytes at a time, LSB first and MSB next. For example, to transmit the word 0x08AA to the device, transmit AA first, followed by 08. The program flow of the CAN bootloader is identical to the SCI bootloader. The data sequence for the CAN bootloader is shown in [Table 4-33](#).

**Table 4-33. CAN 8-Bit Data Stream**

Bytes	Byte 1 of 2	Byte 2 of 2	Description	
1	2	AA	08	0x08AA (KeyValue for memory width = 16 bits)
3	4	00	00	reserved
5	6	00	00	reserved
7	8	00	00	reserved
9	10	00	00	reserved
11	12	00	00	reserved
13	14	00	00	reserved
15	16	00	00	reserved
17	18	00	00	reserved
19	20	BB	AA	Entry point PC[22:16]
21	22	DD	CC	Entry point PC[15:0] (PC = 0xAABB CCDD)
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 the first block Addr[15:0] (Addr = 0xAABB CCDD)
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 second block to load = 0xMMNN words
.		BB	AA	Destination address of the second block Addr[31:16]
.		DD	CC	Destination address of the second block Addr[15:0]
.		BB	AA	First word of the second block in the source being loaded
.				....
n	n+1	BB	AA	Last word of the last block of the source being loaded (More sections if required)
n+2	n+3	00	00	Block size of 0000h - indicates end of the source program

#### 4.7.8 GPIO Assignments

This section details the GPIOs and boot option values used for boot mode set in the BOOT\_DEF memory location located at Z1-OTP-BOOTDEF-LOW/ Z2-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH/ Z2-OTP-BOOTDEF-HIGH. Refer to [Section 4.4.2](#) on how to configure BOOT\_DEFx. When selecting a boot mode option, make sure to verify that the necessary pins are available in the pin mux options for the specific device package being used.

Default boot mode GPIO pins:

- Boot mode pin 0 - GPIO32
- Boot mode pin 1 - GPIO24

Guidelines on boot pin selection:

- Avoid pins that have PWM functionality.
- Can not be analog or USB pins.
- Boot mode select pins and default boot peripheral pins can be available on all packages.
- Avoid JTAG emulation pins and crystal pins.
- Boot mode select pins can be inputs.
- Pins can not have PHY bootstrap functionality.

**Table 4-34. SCI Boot Options**

Option	BOOTDEF Value	SCITXDA GPIO	SCIRXDA GPIO	Package Supported
0 (default)	0x01	GPIO29	GPIO28	All
1	0x21	GPIO1	GPIO0	All
2	0x41	GPIO8	GPIO9	48-QFN, 64-QFP
3	0x61	GPIO7	GPIO3	All
4	0x81	GPIO16	GPIO3	48-QFP, 48-QFN, 64-QFP

**Table 4-35. CAN Boot Options**

Option	BOOTDEF Value	CANTXA GPIO	CANRXA GPIO	Package Supported
0 (default)	0x02	GPIO4	GPIO5	All
1	0x22	GPIO32	GPIO33	48-QFP, 48-QFN, 64-QFP
2	0x42	GPIO2	GPIO3	48-QFP, 48-QFN, 64-QFP
3	0x62	GPIO13	GPIO12	48-QFP, 48-QFN, 64-QFP

**Table 4-36. I2C Boot Options**

Option	BOOTDEF Value	SDAA GPIO	SCLA GPIO	Package Supported
0	0x07	GPIO0	GPIO1	All
1	0x27	GPIO32	GPIO33	48-QFP, 48-QFN, 64-QFP
2	0x47	GPIO5	GPIO4	All

**Table 4-37. SPI Boot Options**

Option	BOOTDEF Value	SPISIMOA	SPISOMIA	SPICLKA	SPISTEA	Package Supported
0	0x06	GPIO7	GPIO1	GPIO3	GPIO5	All
1	0x26	GPIO16	GPIO1	GPIO3	GPIO0	48-QFP, 48-QFN, 64-QFP
2	0x46	GPIO8	GPIO10	GPIO9	GPIO11	64-QFP
3	0x66	GPIO16	GPIO13	GPIO12	GPIO29	48-QFP, 48-QFN, 64-QFP

**Table 4-38. Parallel Boot Options**

Option	BOOTDEF Value	D0-D7 GPIO	C28x (DSP) Control GPIO	Host Control GPIO	Package Supported
0 (default)	0x00	D0 - GPIO0	GPIO224	GPIO242	All
		D1 - GPIO1			
		D2 - GPIO3			
		D3 - GPIO4			
		D4 - GPIO5			
		D5 - GPIO7			
		D6 - GPIO28			
		D7 - GPIO29			
1	0x20	D0 - GPIO0	GPIO12	GPIO13	48-QFP, 48-QFN, 64-QFP
		D1 - GPIO1			
		D2 - GPIO2			
		D3 - GPIO3			
		D4 - GPIO4			
		D5 - GPIO5			
		D6 - GPIO6			
		D7 - GPIO7			
2	0x40	D0 - GPIO0	GPIO16	GPIO29	48-QFP, 48-QFN, 64-QFP
		D1 - GPIO1			
		D2 - GPIO2			
		D3 - GPIO3			
		D4 - GPIO4			
		D5 - GPIO5			
		D6 - GPIO6			
		D7 - GPIO7			

#### 4.7.9 Secure ROM Function APIs

There are a few functions that are available within Secure ROM to be called by the application to perform EXEONLY Flash/RAM tasks in a secure manner.

##### Note

The application can disable interrupts before calling one of the EXEONLY function APIs.

If a vector fetch request is given by the CPU (C28x) while the program counter (PC) is within the EXEONLY function API code of the Secure ROM, a reset fires (RSN if from C28x). The consequence of this is if an NMI or ITRAP or Bus Fault occurs while the PC is executing one of the EXEONLY API functions, the NMI/ITRAP/Fault cannot be serviced because a reset is fired to the subsystem.

The **secure copy code zone 1 function** allow EXEONLY Flash to be copied to EXEONLY RAM in a secure manner. The source must be from EXEONLY Flash and the destination to EXEONLY RAM. There is no support to copy EXEONLY ROM or EXEONLY RAM to RAM. Both Flash and RAM must be set to EXEONLY and configured for the same zone. Additionally, the copy size must not cross over the Flash sector boundary. Any violations of these requirements results in a failure status returned. Upon successful copy of the data, the number of 16-bit words copied is returned.

**Table 4-39. Secure Copy Code Function**

CPU	Function Prototype	Function Parameters	Function Return Value
CPU (C28x)	<code>uint16_t SecureCopyCodeZ1(uint32_t size, uint16_t *dst, uint16_t *src)</code>	<p><i>size</i> : The number of 16-bit words to copy</p> <p><i>dst</i> : The destination memory address in EXEONLY RAM</p> <p><i>src</i> : The source memory address in EXEONLY Flash</p>	<p>0xFFFF : Returns the number of 16-bit words copied.</p> <p>0x0000 : Indicates one of the following: Copy length is zero; Copy size crosses over Flash sector boundary; Flash and RAM do not belong to the same zone; Flash and RAM are not set to EXEONLY; Error occurred during data copy</p>

The **secure CRC calculation zone 1 function** allow a safety CRC check of EXEONLY memory in a secure manner. The CRC length provided must be a value from 1 to 8 where 1 represents a CRC size of 32 16-bit words and 8 represents a CRC size of 4096 16-bit words. The source address specifies the starting address for the CRC and the destination address is the location that the resulting CRC value is stored. The source and destination memories must be configured for the same zone. Additionally, the CRC length must not cross over the Flash sector or RAM block boundary. Any violations of these requirements results in a failure status returned. Upon successful CRC, the number of 16-bit words CRCed is returned.

**Table 4-40. Secure CRC Calculation Function**

CPU	Function Prototype	Function Parameters	Function Return Value
CPU (C28x)	<code>uint16_t SecureCRCCalcZ1(uint16_t len_id, uint16_t *dst, uint16_t *src)</code>	<p><i>len_id</i> : A number from 1 to 8 that corresponds to length options of 32, 64, 128, 256, 512, 1024, 2048, or 4096 16-bit words</p> <p><i>dst</i> : The destination memory address for resulting CRC</p> <p><i>src</i> : The source memory address to begin CRC calculation</p>	<p>0xFFFF : Returns the number of 16-bit words CRCed</p> <p>0x0000 : Indicates one of the following: Invalid length option; Source address is not modulo of length value; Destination address is not within secure RAM; CRC size crosses over Flash sector or RAM block boundary; The source and destination memory do not belong to the same zone.</p>

The **CMAC calculate and compare function** allows to calculate CMAC signature of a Flash memory block and compare against a golden signature. This is used in the secure boot mode to authenticate the boot image.

**Table 4-41. CMAC Calculation Function**

CPU	Function Prototype	Function Parameters	Function Return Value
CPU (C28x)	<code>uint32_t CPU1BROM_calculateCMAC(uint32_t startAddress, uint32_t endAddress, uint32_t signatureAddress)</code>	<b>startAddress:</b> Starting address of memory for which CMAC has to be calculated <b>endAddress:</b> Ending address of memory for which CMAC has to be calculated <b>signatureAddress:</b> Address of location where golden CMAC signature is stored	<b>0xFFFF FFFFU:</b> Calculated CMAC signature did not match golden signature (fail) <b>0xA5A5 A5A5U:</b> Memory range provided is not aligned to 128-bit boundary or length is zero <b>0xE1E1 E1E1U:</b> AES Engine timed out <b>0x0000 0000U:</b> No Error

#### 4.7.10 Clock Initializations

During boot-up, the boot ROM initializes the device clocking, depending upon the reset source, to assist in faster boot time response. Clock configurations are performed by the boot ROM code only for POR and XRS reset types. For all other resets, the boot ROM starts executing with the clocks that were already set up before reset.

#### Note

CPU performs clock configurations during boot up. If the PLL is used during the boot process, the PLL is bypassed by the boot ROM code before branching to the user application.

**Table 4-42. CPU Boot Clock Sources**

Source	Frequency	Description
INTOSC2	10MHz	Default clock source
INTOSC1	10MHz	Set as clock source if missing clock is detected at power up or right after device reset
SYSPLL	95MHz, 47.5MHz	Enabled optionally as part of main boot flow or as part of MPOST POR memory test boot flow. PLL is bypassed and disabled after memory test has completed. See more details regarding enabling MPOST POR memory test in <a href="#">Section 4.7.11.2</a> .

**Table 4-43. CPU Clock State After Boot**

Reset Source	Clock State
POR/XRS	1. Using INTOSC2 2. System clock divider set to /1
All other Resets	Maintain clocks setup before device reset.

#### 4.7.11 Boot Status Information

Boot ROM keeps a record of the various actions and events that occur during boot ROM execution. The reason for this is because NMI and other exceptions are enabled by default in the device and must be handled accordingly. Boot ROM stores the boot status information in a RAM location so that the user application can read this boot status and take the necessary actions per application's needs to handle these events.

##### 4.7.11.1 Booting Status

Boot ROM health and booting status is written to a 32-bit address in M0RAM. This status is cleared on a POR or XRS reset. The previous status is retained on any other reset. For example, you should clear the status before performing a debugger device reset in order to view the latest boot ROM actions reflected in the status.

**Table 4-44. Boot Status Address**

Description	Address
Boot ROM Status	0x0000 0002

**Table 4-45. Boot Status Bit Fields**

Value	Description
0x4000 0000	Flash 2T Not Ready
0x2000 0000	TRIM Load Error
0x1000 0000	RAM Init Error
0x0800 0000	Flash Verification Error
0x0400 0000	DCSM initialization LP Error
0x0200 0000	DCSM Initialization Invalid LP
0x0100 0000	SYSPLL enabled successfully
0x0040 0000	Missing clock NMI occurred
0x0020 0000	RAM Uncorrectable Error NMI occurred
0x0010 0000	Flash Uncorrectable Error NMI occurred
0x0008 0000	RL NMI occurred
0x0004 0000	ERAD NMI occurred
0x0002 0000	Boot ROM detected a PIE mismatch
0x0001 0000	Boot ROM detected an ITRAP
0x0000 8000	Boot ROM has completed running
0x0000 4000	Watchdog self test fail
0x0000 2000	Boot ROM handled POR
0x0000 1000	Boot ROM handled XRS
0x0000 0800	Boot ROM handled all the resets
0x0000 0400	POR memory test has completed
0x0000 0200	DCSM initialization has completed
0x0000 0100	RAM Initialization Complete
0x0000 000B	Wait boot has started
0x0000 0009	CAN boot has started
0x0000 0008	I2C boot has started
0x0000 0007	SPI boot has started
0x0000 0006	SCI boot has started
0x0000 0005	RAM boot has started
0x0000 0004	Parallel boot has started
0x0000 0003	Secure Flash boot has started
0x0000 0002	Flash boot has started
0x0000 0001	Boot ROM has started running

#### 4.7.11.2 Boot Mode and MPOST (Memory Power On Self-Test) Status

Once the boot mode is decoded during the boot flow, the boot mode value is written to RAM. Additionally when running the MPOST POR memory test, the test result is written to RAM.

For more information, see the [C2000™ Memory Power-On Self-Test \(M-POST\) Application Report](#).

**Table 4-46. Boot Mode and MPOST Status Addresses**

Description	Address
Boot Mode	0x0000 0004
MPOST POR Memory Test Result	0x0000 0006

#### 4.7.12 ROM Version

The ROM revision and release date information is stored at the ROM locations specified in [Table 4-47](#). Reading a revision number value of “0x100” represents version “1.0”, “0x101” represents version “1.1”, and so on. Reading a revision date value of “0x0119” represents “01/19” or “January 2019”.

**Table 4-47. Boot ROM Version Information**

Contents	Address
Revision Number	0x003F 9804
Revision Date	0x003F 9805
Build Number	0x003F 9806

### 4.8 Application Notes for Using the Bootloaders

#### 4.8.1 Bootloader Data Stream Structure

This section details the data transfer protocols or stream structures that allow boot data transfer between boot ROM and host device. This data transfer protocol is compatible to the respective bootloaders on C2000 devices.

[Table 4-48](#) and [Example 4-2](#) 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. Refer to [Section 4.8.2](#) for more details on using the C28x hex utility to convert a project to this format.

The first 16-bit word in the data stream is known as the key value. The key value is used to indicate to the bootloader the width of the incoming stream: 8 or 16 bits. Note that not all bootloaders accept both 8- and 16-bit streams. 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; for a 16-bit data stream, the key value 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 the bootloader. If a bootloader does not use these values then the values are reserved for future use and the bootloader simply reads the value and then discards the value. Currently only the SPI and I2C and parallel bootloaders use these words to initialize registers.

The tenth and eleventh words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the bootloader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined as 8-bit data stream format. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size is 0x000A to indicate 10 16-bit words.

The next two words indicate to the loader the destination address of the block of data. Following the size and address is the 16-bit words that makeup that block of data.

This pattern of block size/destination address repeats for each block of data to be transferred. Once all the blocks have been transferred, a block size of 0x0000 signals to the loader that the transfer is complete. At this point, the loader returns the entry point address to the calling routine, which cleans up and exits. Execution then continues at the entry point address as determined by the input data stream contents.

**Table 4-48. LSB/MSB Loading Sequence in 8-Bit Data Stream**

		Contents	
Byte		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 indicates 10 words or 20 bytes in the block.	MSB: block size
25	26	LSB: MSW destination address, first block Addr[23:16]	MSB: MSW destination address, first block Addr[31:24]
27	28	LSB: LSW destination address, first block Addr[7:0]	MSB: LSW destination address, first block Addr[15:8]
29	30	LSB: First word of the first block being loaded	MSB: First word of the first block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the first block to load	MSB: Last word of the first block to load
.	.	LSB: Block size of the second block	MSB: Block size of the second block
.	.	LSB: MSW destination address, second block Addr[23:16]	MSB: MSW destination address, second block Addr[31:24]
.	.	LSB: LSW destination address, second block Addr[7:0]	MSB: LSW destination address, second block Addr[15:8]
.	.	LSB: First word of the second block being loaded	MSB: First word of the second block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the second block	MSB: Last word of the second block
.	.	LSB: Block size of the last block	MSB: Block size of the last block
.	.	LSB: MSW of destination address of last block Addr[23:16]	MSB: MSW destination address, last block Addr[31:24]
.	.	LSB: LSW destination address, last block Addr[7:0]	MSB: LSW destination address, last block Addr[15:8]
.	.	LSB: First word of the last block being loaded	MSB: First word of the last block being loaded
...	...	...	...
...	...	...	...
.	.	LSB: Last word of the last block	MSB: Last word of the last block
n	n+1	LSB: 00h	MSB: 00h - indicates the end of the source

### Example 4-2. Data Stream Structure 8-bit

```

AA 08      ; 0x08AA 8bit 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 is 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 is 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 are initialized as follows:
Location Value
0x3F9010 0x0001
0x3F9011 0x0002
0x3F9012 0x0003
0x3F9013 0x0004
0x3F9014 0x0005
0x3F8000 0x7700
0x3F8001 0x7625
PC Begins execution at 0x3F8000
  
```

## 4.8.2 The C2000 Hex Utility

To use the features of the bootloader, you must generate a data stream and boot table as described in [Section 4.8.1](#). The hex conversion utility tool, included with the C28x code generation tools, can generate the required data stream including the required boot table. This section describes the hex2000 utility. An example of a file conversion performed by hex2000 is described in [Example 4-3](#).

The hex utility supports creation of the boot table required for the SCI, SPI, I2C, CAN, and parallel I/O loaders. That is, the hex utility adds the required information to the file such as the key value, reserved bits, entry point, address, block start address, block length and terminating value. The contents of the boot table vary slightly depending on the boot mode and the options selected when running the hex conversion utility. The actual file format required by the host (ASCII, binary, hex, and so on) differs from one specific application to another and some additional conversion can be required.

To build the boot table, follow these steps:

1. **Assemble or compile the code.** This creates the object files that are then used by the linker to create a single output file.
2. **Link the file.** The linker combines all of the object files into a single output file in common object file format (ELF). The specified linker command file is used by the linker to allocate the code sections to different memory blocks. Each block of the boot table data corresponds to an initialized section in the ELF file. Uninitialized sections are not converted by the hex conversion utility. The following options can be useful:
  - The linker `-m` option can be used to generate a map file. This map file shows all of the sections that were created, the location in memory, and the length. Check this file to make sure that the initialized sections are where you expect them.
  - The linker `-w` option configures the linker to show, if the linker assigned a section to a memory region automatically. For example, if you have a section in your code called `.TI.ramfunc`.
3. **Run the hex conversion utility.** Choose the appropriate options for the desired boot mode and run the hex conversion utility to convert the ELF file produced by the linker to a boot table.

See the [TMS320C28x Assembly Language Tools User's Guide](#) and the [TMS320C28x Optimizing C/C++ Compiler User's Guide](#) for more information on the compiling and linking process.

**Table 4-49** summarizes the hex conversion utility options available for the bootloader. See the [TMS320C28x Assembly Language Tools User's Guide](#) for a detailed description of the hex2000 operations used to generate a boot table. Updates are made to support the I2C boot. See the Codegen release notes for the latest information.

**Table 4-49. Boot Loader Options**

Option	Description
-boot	Convert all sections into bootable form (use instead of a SECTIONS directive)
-sci8	Specify the source of the bootloader table as the SCI-A port, 8-bit mode
-spi8	Specify the source of the bootloader table as the SPI-A port, 8-bit mode
-gpio8	Specify the source of the bootloader table as the GPIO port, 8-bit mode
-bootorg value	Specify the source address of the bootloader table
-lospcp value	Specify the initial value for the LOSPCP register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-spibrr value	Specify the initial value for the SPIBRR register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
-e value	Specify the entry point at which to begin execution after boot loading. The value can be an address or a global symbol. This value is optional. The entry point can be defined at compile time using the linker -e option to assign the entry point to a global symbol. The entry point for a C program is normally <code>_c_int00</code> unless defined otherwise by the -e linker option.
-i2c8	Specify the source of the bootloader table as the I2C-A port, 8-bit
-i2cpsc value	Specify the value for the I2CPSC register. This value is loaded and takes effect after all I2C options are loaded, prior to reading data from the EEPROM. This value is truncated to the least-significant eight bits and can be set to maintain an I2C module clock of 7 to 12MHz.
-i2cclk value	Specify the value for the I2CCLKH register. This value is loaded and takes effect after all I2C options are loaded, prior to reading data from the EEPROM.
-i2cclk value	Specify the value for the I2CCLKL register. This value is loaded and takes effect after all I2C options are loaded, prior to reading data from the EEPROM.

### Example 4-3. HEX2000.exe Command Syntax

```
C: HEX2000 GPIO34TOG.OUT -boot -gpio8 -a
where:
- boot Convert all sections into bootable form.
- gpio8 Use the GPIO in 8-bit mode data format. The eCAN
        uses the same data format as the GPIO in 8bit mode.
- a     Select ASCII-Hex as the output format.
```

This page intentionally left blank.

Chapter 5  
**Dual Code Security Module (DCSM)**

---



This chapter explains the dual code security module.

<b>5.1 Introduction</b> .....	<b>396</b>
<b>5.2 Functional Description</b> .....	<b>396</b>
<b>5.3 Flash and OTP Erase/Program</b> .....	<b>403</b>
<b>5.4 Secure Copy Code</b> .....	<b>403</b>
<b>5.5 SecureCRC</b> .....	<b>404</b>
<b>5.6 CSM Impact on Other On-Chip Resources</b> .....	<b>405</b>
<b>5.7 Incorporating Code Security in User Applications</b> .....	<b>407</b>
<b>5.8 Software</b> .....	<b>412</b>
<b>5.9 DCSM Registers</b> .....	<b>412</b>

## 5.1 Introduction

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

The CSM has dual-zone security: Zone1 (Z1) and Zone2 (Z2).

### 5.1.1 DCSM Related Collateral

#### Getting Started Materials

- [C2000 DCSM Security Tool Application Report](#)
- [C2000 Unique Device Number Application Report](#)
- [Enhancing Device Security by Using JTAGLOCK Feature Application Report](#)
- [Secure BOOT On C2000 Device Application Report](#)

## 5.2 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.

The code security mechanism offers protection for two zones, Zone1 (Z1) and Zone2 (Z2). The security mechanism for both the zones is identical. Each zone has a dedicated secure resource and allocated secure resource. The following are different secure resources available on this device:

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

[Table 5-1](#) shows the status of a RAM block/Flash sector based on the configuration in the GRABRAM/GRABSECT register.

The security of each zone is enabled by a 128-bit (four 32-bit words) password (CSM password). The password for each zone is stored in USER OTP. A zone can be unsecured by executing the password match flow (PMF), described in [Section 5.7.4](#).

There are three types of accesses:

- **Data/program reads:** Data reads to secure memory are always blocked unless the program is executing from a memory that belongs to the same zone. Data reads to unsecure memory are always allowed. [Table 5-2](#) shows the levels of security.
- **JTAG access:** JTAG accesses are always blocked when a memory is secure.
- **Instruction fetches (calls, jumps, code executions, ISRs):** Instruction fetches are never blocked.

**CAUTION**

Never program any other values in these fields. Failing any of these conditions for a RAM block/Flash sector makes that RAM block/Flash sector inaccessible.

**Table 5-1. RAM/Flash Status**

Zone 1 GRABRAMx/GRABSECTx Bits	Zone 2 GRABRAMx/GRABSECTx Bits	Ownership and Accessibility
01	10	RAM block/Flash Sector belongs to Zone1
01	11 <sup>(2)</sup>	RAM block/Flash Sector belongs to Zone1
10	01	RAM block/Flash Sector belongs to Zone2
11 <sup>(1)</sup>	01	RAM block/Flash Sector belongs to Zone2
10	10	RAM block/Flash Sector is unsecure
11	11	RAM block/Flash Sector inaccessible if either of the zones is secure (CSM passwords are programmed). User must never leave these values default (11), if CSM passwords are programmed for even one zone.

- (1) Zone1 must be unsecure. Assumption in this case is that user is not using Zone1 so none of the fields, including passwords, in Zone1 USER OTP are programmed by user; hence, Zone1 is always unsecure.
- (2) Zone2 must be unsecure. Assumption in this case is that user is not using Zone2 so none of the fields, including passwords, in Zone2 USER OTP are programmed by user; hence, Zone2 is always unsecure.

**Table 5-2. Security Levels**

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

**5.2.1 CSM Passwords**

Unlike earlier C2000™ devices, on this device ALL\_1 value (0x FFFF FFFF, 0x FFFF FFFF, 0x FFFF FFFF, 0x FFFF FFFF) for CSM password for a zone does not unsecure the zone. Instead, if for any zone the CSM password values get loaded as ALL\_1 from USER OTP, the device is in BLOCKED state. Due to this reason, TI programs a few bits in the second 32-bit password value (ZxOTP\_CSMPSWD1) in every zone select block of each zone with value 0. The default value for this password location is chosen in a manner that the respective ECC value remains ALL\_1. Due to this, the CSMPSWD1 value programmed by TI for every zone select block is different. See [Table 5-3](#) for ZxOTP\_CSMPSWD1 value, programmed by TI on every device. Since ECC is not programmed, the user is able to change this value by flipping the bits that are 1 to 0 but leaving the bits that are already programmed by TI as 0. BOOTROM code writes the default password value into the KEYx register to unlock the device as part of device initialization sequence.

If the password locations of a zone have all 128 bits as zeros (ALL\_0), that zone becomes permanently secure (LOCKED state), regardless of the contents of the CSMKEYx registers, which means the zone cannot be unlocked using PMF, see the password match flow described in [Section 5.7.2](#). Therefore, the user can never use ALL\_0 as password. A password of ALL\_0 prevents debug of secure code or reprogramming the Flash sectors. CSMKEYx registers are user-accessible registers that are used to unsecure the zones.

**Table 5-3. Default Value of ZxOTP (Programmed by TI)**

Zone Select Block	Zone1 USER OTP		Zone2 USER OTP	
	Address	Value	Address	Value
JLM_ENABLE (JTAGLOCK)	0x0007 8006	0xFFFF 000F	NA	NA
PSWDLOCK	0x0007 8010	0xFB7F FFFF	0x0007 8210	0x1F7F FFFF
CRCLOCK	0x0007 8012	0x7FFF FFFF	0x0007 8212	0x3FFF FFFF
JTAGPSWDH0	0x0007 8014	0x4BFF FFFF	NA	NA
JTAGPSWDH1	0x0007 8016	0x3FFF FFFF	NA	NA
Zone_Select_Block0	0x0007 8022 (CSMPSWD1)	0x4D7F FFFF	0x0007 8222 (CSMPSWD1)	0x1F7F FFFF
Zone_Select_Block0	0x0007 803E (JTAGPSWDL1)	0x2BFF FFFF	NA	NA
Zone_Select_Block1	0x0007 8042 (CSMPSWD1)	0x5F7F FFFF	0x0007 8242 (CSMPSWD1)	0xE57F FFFF
Zone_Select_Block1	0x0007 805E (JTAGPSWDL1)	0x27FF FFFF	NA	NA
Zone_Select_Block2	0x0007 8062 (CSMPSWD1)	0x1DFF FFFF	0x0007 8262 (CSMPSWD1)	0x4FFF FFFF
Zone_Select_Block2	0x0007 807E (JTAGPSWDL1)	0x7B7F FFFF	NA	NA
Zone_Select_Block3	0x0007 8082 (CSMPSWD1)	0xAF7F FFFF	0x0007 8282 (CSMPSWD1)	0xE37F FFFF
Zone_Select_Block3	0x0007 809E (JTAGPSWDL1)	0xC9FF FFFF	NA	NA
Zone_Select_Block4	0x0007 80A2 (CSMPSWD1)	0x1BFF FFFF	0x0007 82A2 (CSMPSWD1)	0x57FF FFFF
Zone_Select_Block4	0x0007 80BE (JTAGPSWDL1)	0x7D7F FFFF	NA	NA
Zone_Select_Block5	0x0007 80C2 (CSMPSWD1)	0x17FF FFFF	0x0007 82C2 (CSMPSWD1)	0x5BFF FFFF
Zone_Select_Block5	0x0007 80DE (JTAGPSWDL1)	0x6F7F FFFF	NA	NA
Zone_Select_Block6	0x0007 80E2 (CSMPSWD1)	0xBD7F FFFF	0x0007 82E2 (CSMPSWD1)	0xF17F FFFF
Zone_Select_Block6	0x0007 80FE (JTAGPSWDL1)	0x33FF FFFF	NA	NA
Zone_Select_Block7	0x0007 8102 (CSMPSWD1)	0x9F7F FFFF	0x0007 8302 (CSMPSWD1)	0x3B7F FFFF
Zone_Select_Block7	0x0007 811E (JTAGPSWDL1)	0x0FFF FFFF	NA	NA
Zone_Select_Block8	0x0007 8122 (CSMPSWD1)	0x2BFF FFFF	0x0007 8322 (CSMPSWD1)	0x8FFF FFFF
Zone_Select_Block8	0x0007 813E (JTAGPSWDL1)	0xBB7F FFFF	NA	NA
Zone_Select_Block9	0x0007 8142 (CSMPSWD1)	0x27FF FFFF	0x0007 8342 (CSMPSWD1)	0x6BFF FFFF
Zone_Select_Block9	0x0007 815E (JTAGPSWDL1)	0x5F7F FFFF	NA	NA
Zone_Select_Block10	0x0007 8162 (CSMPSWD1)	0x7B7F FFFF	0x0007 8362 (CSMPSWD1)	0x377F FFFF
Zone_Select_Block10	0x0007 817E (JTAGPSWDL1)	0x1DFF FFFF	NA	NA
Zone_Select_Block11	0x0007 8182 (CSMPSWD1)	0xC9FF FFFF	0x0007 8382 (CSMPSWD1)	0x9BFF FFFF
Zone_Select_Block11	0x0007 819E (JTAGPSWDL1)	0xAF7F FFFF	NA	NA
Zone_Select_Block12	0x0007 81A2 (CSMPSWD1)	0x7D7F FFFF	0x0007 83A2 (CSMPSWD1)	0x2F7F FFFF
Zone_Select_Block12	0x0007 81BE (JTAGPSWDL1)	0x1BFF FFFF	NA	NA
Zone_Select_Block13	0x0007 81C2 (CSMPSWD1)	0x6F7F FFFF	0x0007 83C2 (CSMPSWD1)	0xCB7F FFFF
Zone_Select_Block13	0x0007 81DE (JTAGPSWDL1)	0x17FF FFFF	NA	NA
Zone_Select_Block14	0x0007 81E2 (CSMPSWD1)	0x33FF FFFF	0x0007 83E2 (CSMPSWD1)	0x97FF FFFF
Zone_Select_Block14	0x0007 81FE (JTAGPSWDL1)	0xBD7F FFFF		

### 5.2.2 Emulation Code Security Logic (ECSL)

In addition to the CSM, the emulation code security logic (ECSL) has been implemented using a 64-bit password (part of existing CSM password) for each zone to prevent unauthorized users from stepping through secure code. A halt in secure code while the emulator is connected trips the ECSL and break the emulation connection. To allow emulation of secure code, while maintaining the CSM protection against secure memory reads, you 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 disables 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 starts running and can 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 trips and causes the emulator connection to be broken.

One answer to this problem is to use the Wait Boot Mode boot option. In this mode, the CPU is in a loop and does not jump to the user application code. Using this BOOTMODE, you can connect to CCS and debug the code.

### 5.2.3 CPU Secure Logic

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

### 5.2.4 Execute-Only Protection

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

### 5.2.5 Password Lock

The password locations in USER OTP 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 are not 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 new device, the value for password lock fields for all zones at the PSWDLOCK location in OTP is 1111, which means the password for all zones is unlocked.

---

#### Note

Password unlock only makes password locations non-secure. All other secure memories remains secure as per security settings. Since password locations are non-secure, anyone can read the password and make the zone unsecure by running through PMF, the user must program the PSWDLOCK locations to lock the password before sending the device in the field.

---

### 5.2.6 JTAGLOCK

To disable the JTAG access on a device to avoid any debug access, use the JTAGLOCK feature on this device. Follow a two step process to enable the JTAGLOCK feature (both steps can be performed at the same time):

1. Program the JTAG passwords. This device has a 128-bit JTAG password that needs to be programmed in Z1 USER OTP. JTAG passwords are split into two parts, JTAGPSWDH and JTAGPSWDL. JTAGPSWDH is part of Z1 USER OTP header and JTAGPSWDL is part of Z1 Zone Select Block (ZSB). What this means is program JTAGPSWDH once and change the JTAGPSWDL multiple times, if needed. The Code Composer Studio™ IDE has an integrated tool that you use to unlock the JTAGLOCK on the device.
2. After programming the JTAG passwords, enable the JTAGLOCK module (JLM) by programming bit [3:0] of Z1OTP\_JLM\_ENABLE with any value other than 0xF. It is recommended to program all four bits with a value 0x0.

### 5.2.7 Link Pointer and Zone Select

For each of the two security zones, a dedicated OTP block exists that holds the configuration related to zone's security. The following are user programmable configurations:

- ZxOTP\_LINKPOINTER1
- ZxOTP\_LINKPOINTER2
- ZxOTP\_LINKPOINTER3
- Z1OTP\_JLM\_ENABLE
- ZxOTP\_GPREG1
- ZxOTP\_GPREG2
- ZxOTP\_GPREG3
- ZxOTP\_GPREG4
- ZxOTP\_PSWDLOCK
- ZxOTP\_CRCLOCK
- Z1OTP\_JTAGPSWDH
- Z1OTP\_CMACKKEY
- ZxOTP\_CSMPSWD0
- ZxOTP\_CSMPSWD1
- ZxOTP\_CSMPSWD2
- ZxOTP\_CSMPSWD3
- ZxOTP\_GRABSECT1
- ZxOTP\_GRABSECT2
- ZxOTP\_GRABSECT3
- ZxOTP\_GRABRAM1
- ZxOTP\_EXEONLYSECT1
- ZxOTP\_EXEONLYSECT2
- ZxOTP\_EXEONLYRAM1
- Z1OTP\_JTAGPSWDL

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

- ZxOTP\_CSMPSWD0
- ZxOTP\_CSMPSWD1
- ZxOTP\_CSMPSWD2
- ZxOTP\_CSMPSWD3
- ZxOTP\_GRABSECT1
- ZxOTP\_GRABSECT2
- ZxOTP\_GRABSECT3
- ZxOTP\_GRABRAM1
- ZxOTP\_EXEONLYSECT1
- ZxOTP\_EXEONLYSECT2
- ZxOTP\_EXEONLYRAM1
- Z1OTP\_JTAGPSWDL

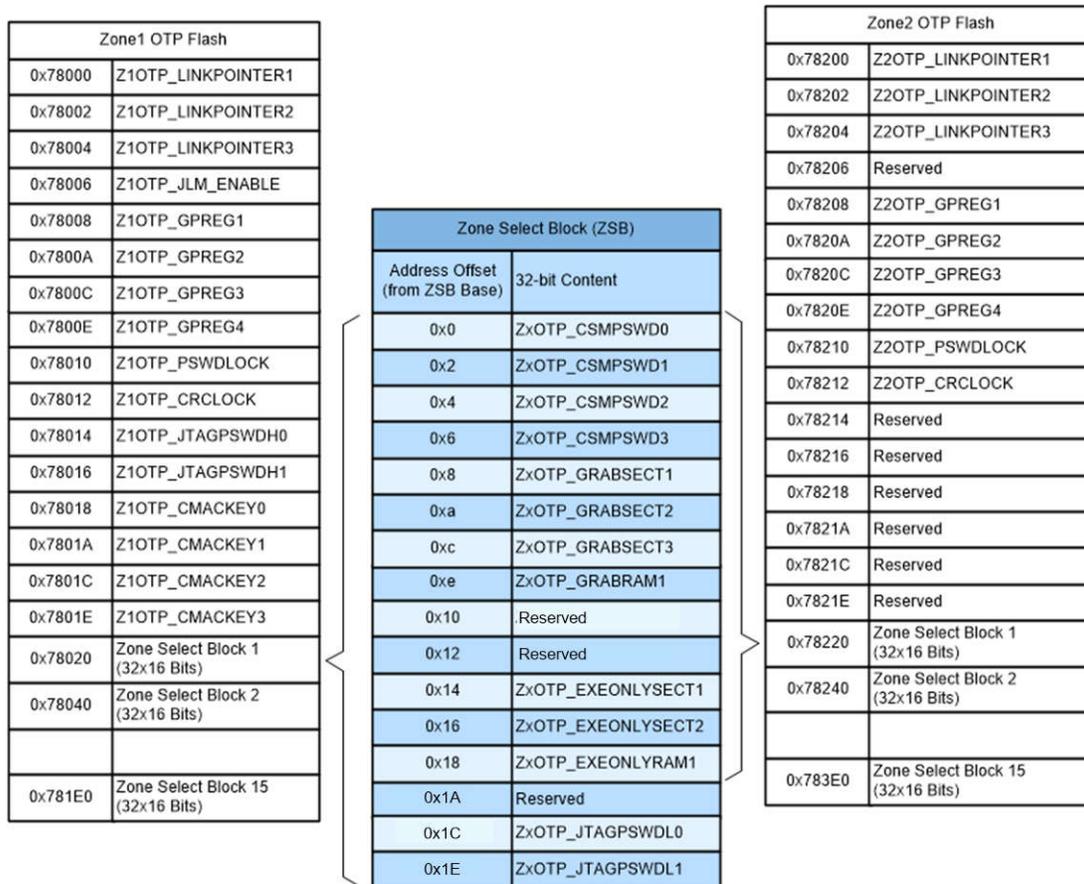
The location of the valid zone select block in OTP is decided based on the value of three 14-bit link pointers (Zx-LINKPOINTERx) programmed in the OTP of each zone. All OTP locations except link pointers and Z1OTP\_JLM\_ENABLE locations are protected with ECC. Since the link pointer locations are not 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 a 0 cannot be flipped to a 1 (no erase operation for OTP), the most-significant bit position in the resolved link pointer that is 0 defines the valid base address for the zone select block. While generating the final link pointer value, if the bit pattern is not one of those listed in [Figure 5-1](#), the final link pointer value becomes All\_1 (0xFFFF\_FFFF), which selects the Zone-Select-Block1 (also known as the default zone select block).

Zx-LINKPOINTER	Selected ZSB	Zone1 ZSB Address	Zone2 ZSB Address
32xxxxxxxxxxxxxxxxxxxx11111111111111	ZSB1	0x78020	0x78220
32xxxxxxxxxxxxxxxxxxxx11111111111110	ZSB2	0x78040	0x78240
32xxxxxxxxxxxxxxxxxxxx11111111111100	ZSB3	0x78060	0x78260
32xxxxxxxxxxxxxxxxxxxx111111111111000	ZSB4	0x78080	0x78280
32xxxxxxxxxxxxxxxxxxxx1111111111110000	ZSB5	0x780a0	0x782a0
32xxxxxxxxxxxxxxxxxxxx11111111111100000	ZSB6	0x780c0	0x782c0
32xxxxxxxxxxxxxxxxxxxx111111111111000000	ZSB7	0x780e0	0x782e0
32xxxxxxxxxxxxxxxxxxxx1111111100000000	ZSB8	0x78100	0x78300
32xxxxxxxxxxxxxxxxxxxx1111111000000000	ZSB9	0x78120	0x78320
32xxxxxxxxxxxxxxxxxxxx1111100000000000	ZSB10	0x78140	0x78340
32xxxxxxxxxxxxxxxxxxxx1111000000000000	ZSB11	0x78160	0x78360
32xxxxxxxxxxxxxxxxxxxx1110000000000000	ZSB12	0x78180	0x78380
32xxxxxxxxxxxxxxxxxxxx1100000000000000	ZSB13	0x781a0	0x783a0
32xxxxxxxxxxxxxxxxxxxx1000000000000000	ZSB14	0x781c0	0x783c0
32xxxxxxxxxxxxxxxxxxxx0000000000000000	ZSB15	0x781e0	0x783e0

**Figure 5-1. Storage of Zone-Select Bits in OTP**

**Note**

Address locations for other security settings that are not part of Zone Select blocks can be programmed only once; therefore, you must program the address locations toward the end of the development cycle.


**Figure 5-2. Location of Zone-Select Block Based on Link-Pointer**
**CAUTION**

USER OTP is ECC protected. You must program the ECC value while programming the security setting in USER OTP. Failing to program the correct ECC value causes the device to be blocked permanently and the device needs to be replaced.

### 5.2.8 C Code Example to Get Zone Select Block Addr for Zone1

```

unsigned long LinkPointer;
unsigned long *ZoneSelBlockPtr;
int Bitpos = 13;
int ZeroFound = 0;
// Read Z1-Linkpointer register of DCSM module.
LinkPointer = *(unsigned long *)0x5F000;
// Bits 31 to 15 as most-significant 0 are reserved LinkPointer options
LinkPointer = LinkPointer << 18;
while ((ZeroFound == 0) && (bitpos > -1))
{
    if ((LinkPointer & 0x80000000) == 0)
    {
        ZeroFound = 1;
        ZoneSelBlockPtr = (unsigned long *) (0x78000 + ((bitpos + 2)*32));
    }
    else
    {
        bitpos--;
        LinkPointer = LinkPointer << 1;
    }
}
if (ZeroFound == 0)
{
    //Default in case there is no zero found.
    ZoneSelBlockPtr = (unsigned long *)0x78020;
}

```

### 5.3 Flash and OTP Erase/Program

On this device, OTP as well as normal Flash, are secure resources. Each zone has a dedicated OTP, whereas normal Flash sectors can be allocated to any zone based on the value programmed in the GRABSECTx location in OTP. Each zone has 128-bit CSM passwords. Read and write accesses are not allowed to resources assigned to Z1 by code running from memory allocated to Z2 and conversely. 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 that belongs to the same zone. The same is the case for erasing any secure Flash sector. To program the security settings in OTP Flash, the user must unlock the CSM of the respective zone. Unless the zone is unlocked, security settings in OTP Flash can not be updated. The OTP content cannot be erased.

A semaphore mechanism is provided to avoid the program/erase conflict between Z1 and Z2. A zone needs to grab this semaphore to successfully complete the erase/program operation on the secure 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.

### 5.4 Secure Copy Code

In some applications, the user wants 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 “Secure Copy Code” library functions for ZONE1 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.

## 5.5 SecureCRC

Since reads from EXEONLY memories are not allowed, the user cannot calculate the CRC on content in EXEONLY memories using the CRC engine available on this device (for example, VCUCRC, GCRC) or software. In some safety-critical applications, the user has to calculate the CRC even on these memories. To enable this without compromising on security, TI provides specific “SecureCRC” 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 is modulo the number of words (based on length\_id) for which the CRC needs to be calculated.
- The destination address belongs 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 secure functions in ROM. If there is a vector fetch during secure function execution, the CPU gets reset immediately.

---

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

## 5.6 CSM Impact on Other On-Chip Resources

### CAUTION

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

### Note

Security initialization is done by BOOTROM code on all the resets (as part of device initialization) that assert SYSRSn. This is not part of the user application code.

On this device, some of the 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 by CPU after reset (any type of reset) to initialize the security on device.

### Security Initialization

- Dummy Read to address location of SECD (0x703F0, TI-reserved register) in TI OTP
- Dummy Read to address location of Z1OTP\_LINKPOINTER1 in Z1 OTP
- Dummy Read to address location of Z1OTP\_LINKPOINTER2 in Z1 OTP
- Dummy Read to address location of Z1OTP\_LINKPOINTER3 in Z1 OTP
- Dummy Read to address location of Z2OTP\_LINKPOINTER1 in Z2 OTP
- Dummy Read to address location of Z2OTP\_LINKPOINTER2 in Z2 OTP
- Dummy Read to address location of Z2OTP\_LINKPOINTER3 in Z2 OTP
- Dummy Read to address location Z1OTP\_JLM\_ENABLE in Z1 OTP
- Dummy Read to address location of Z1OTP\_GPREG1, Z1OTP\_GPREG2, Z1OTP\_GPREG3, Z1OTP\_GPREG4 in Z1 OTP
- Dummy Read to address location of Z1OTP\_PSWDLOCK in Z1 OTP
- Dummy Read to address location of Z1OTP\_CRCLOCK in Z1 OTP
- Dummy Read to address location of Z1OTP\_JTAGPSWDH0, Z1OTP\_JTAGPSWDH1 in Z1 OTP
- Dummy Read to address location of Z2OTP\_GPREG1, Z2OTP\_GPREG2, Z2OTP\_GPREG3, Z2OTP\_GPREG4 in Z2 OTP
- Dummy Read to address location of Z2OTP\_PSWDLOCK in Z2 OTP
- Dummy Read to address location of Z2OTP\_CRCLOCK in Z2 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 Z1OTP\_GRABSECT1, Z1OTP\_GRABSECT2, Z1OTP\_GRABSECT3 in Z1 OTP
- Dummy read to address location of Z1OTP\_GRABRAM1
- Dummy read to address location of Z1OTP\_EXEONLYSECT1, Z1OTP\_EXEONLYSECT2 in Z1 OTP
- Dummy read to address location of Z1OTP\_EXEONLYRAM1 in Z1 OTP
- Dummy Read to address location of Z1OTP\_JTAGPSWDL0, Z1OTP\_JTAGPSWDL1 in Z1 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 Z2OTP\_GRABSECT1, Z2OTP\_GRABSECT2, Z2OTP\_GRABSECT3 in Z2 OTP
- Dummy read to address location of Z2OTP\_GRABRAM1
- Dummy read to address location of Z2OTP\_EXEONLYSECT1, Z2OTP\_EXEONLYSECT2 in Z2 OTP
- Dummy read to address location of Z2OTP\_EXEONLYRAM1 in Z2 OTP

### 5.6.1 RAMOPEN

RAMOPEN feature on this device enables a user to unsecure all of the secure RAM blocks without unlocking the zone security. Afterward, the RAMs can be converted back into secure RAMs without issuing reset to device. However, the previous contents of the secure RAM are not maintained.

This is a useful feature for Flash programming tools (for example, CCS Flash plugin, serial Flash programmer, and so on), which need to download Flash APIs and data on RAMs to carry out the Flash program/erase operations to unsecure Flash regions. Since debugger and boot loader writes are considered unsecure, the writes require either unsecure RAM or unlocked secure RAM to download the APIs and data to. However, the user can not always have the password required to unlock one or both of the security zones. This can be particularly problematic in a device with limited RAM where the majority of the RAM used during application runtime is designated as secure RAM.

The RAMOPEN feature allows the Flash programming tools to temporarily convert the secured RAM blocks into unsecured RAM blocks to load the API/Data and after Flash program/erase operation is completed, turn them back into secured RAM blocks.

The feature works as:

- RAMOPEN feature is enabled by setting RAMOPENFRC.SET bit to 1
- This clears the content of all the secure RAM (RAMINIT is performed in hardware).
- After completion of RAMINIT operation, RAMOPENSTAT.RAMOPEN bit get set to 1 (RAMOPEN feature is enabled and all secure RAM blocks are unsecure).
- To disable RAMOPEN feature, set RAMOPENCLR.CLEAR bit to 1.
- This again clears the content of all secure RAM (RAMINIT is performed in hardware).
- After completion of RAMINIT operation, RAMOPENSTAT.RAMOPEN bit gets cleared to 0 (RAMOPEN feature is disabled and all secure RAM blocks are again secure).

## 5.7 Incorporating Code Security in User Applications

Code security is typically not required in the development phase of a project. However, security is needed once a robust code is developed for a zone. Before such a code is programmed in the Flash memory, a CSM password must 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.31) is the action that secures the device). From that time on, access to debug the contents of secure memory by any means (using 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.

### 5.7.1 Environments That Require Security Unlocking

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

- Code development using debuggers (such as Code Composer Studio™ IDE). 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 IDE 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. In custom programming that uses 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 SRAM 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 can 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 5-3](#) 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.

### 5.7.2 CSM Password Match Flow

Password match flow (PMF) is essentially a sequence of four dummy reads from password locations (PWL) followed by four writes (32-bit writes) to CSMKEY(0/1/2/3) registers. [Figure 5-3](#) shows how PMF helps to initialize the security logic registers and disable security logic.

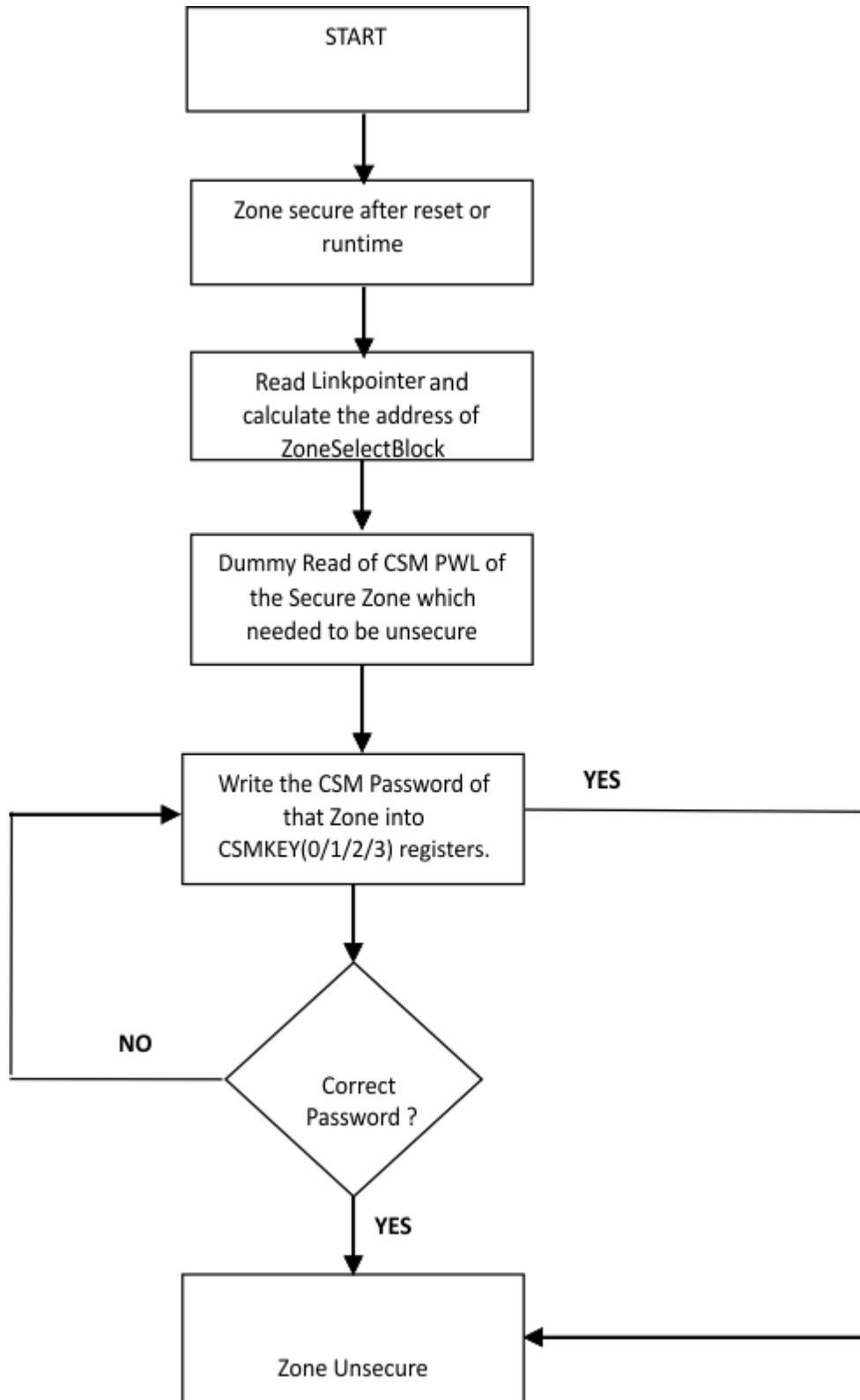


Figure 5-3. CSM Password Match Flow (PMF)

### 5.7.3 C Code Example to Unsecure C28x Zone1

```
volatile long int *CSM = (volatile long int *)5F090; //CSM register file volatile
long int *CSMPWL = (volatile long int *)0x78020; //CSM Password location (assuming default zone
select block)
volatile int tmp;
int I;
// Read the 128-bits of the CSM password locations (PWL)
//
for (I=0;I<4; I++) tmp = *CSMPWL++;
// Write the 128-bit password to the CSMKEY registers
// If this password matches that stored in the
// CSLPWL, then the CSM becomes unsecure. If the password does not
// match, then the zone remains secure.
// An example password of: // 0x11112222333344445555666677778888 is used.
*CSM++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F090
*CSM++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F092
*CSM++ = 0x66665555; // Register Z1_CSMKEY2 at 0x5F094
*CSM++ = 0x88887777; // Register Z1_CSMKEY3 at 0x5F096
```

### 5.7.4 C Code Example to Resecure C28x Zone1

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

### 5.7.5 Environments That Require ECSL Unlocking

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

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

### 5.7.6 ECSL Password Match Flow

A password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by two writes to KEY registers. [Figure 5-4](#) shows how the PMF helps to initialize the security logic registers and disable security logic.

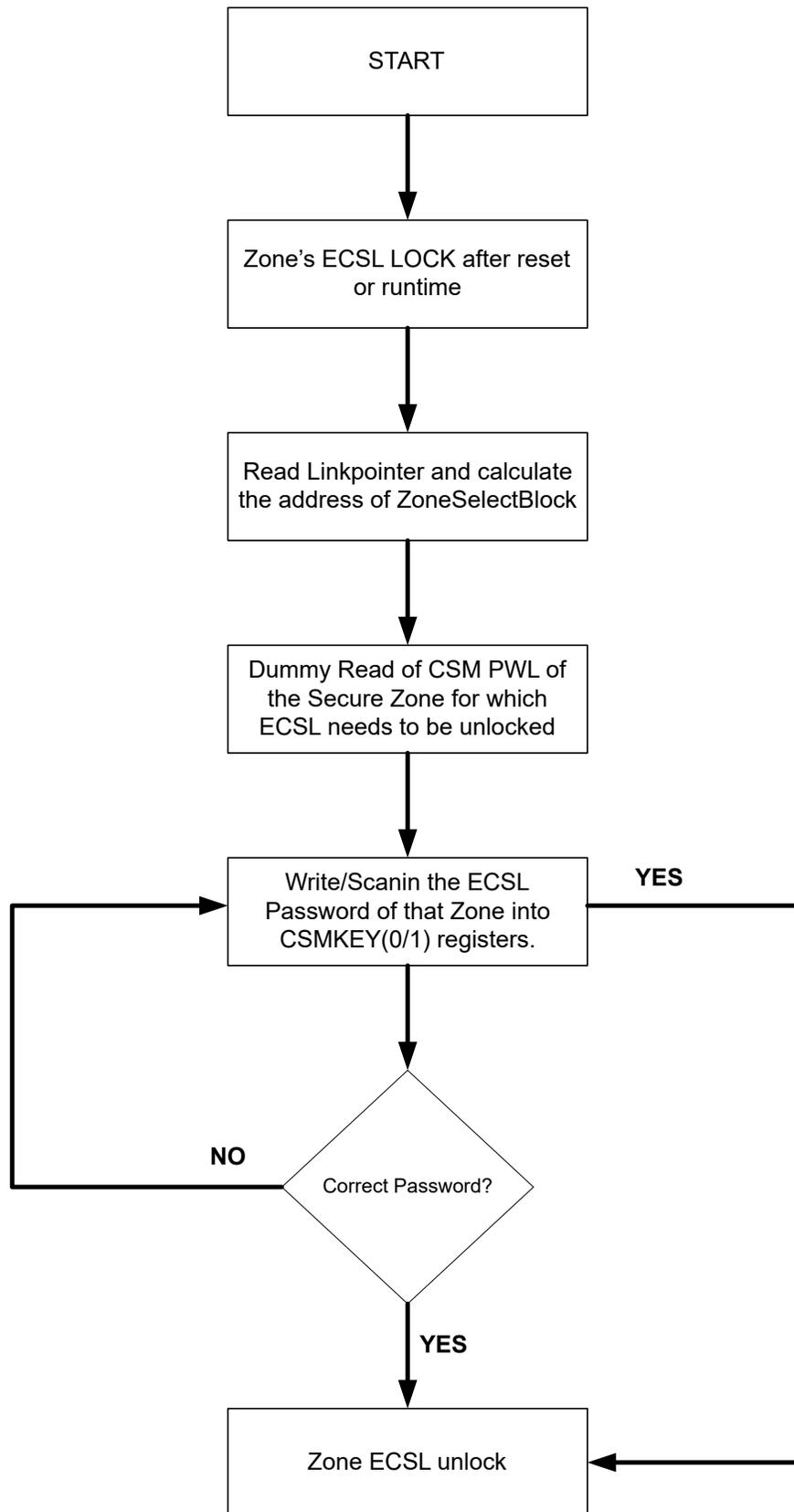


Figure 5-4. ECSSL Password Match Flow (PMF)

### 5.7.7 ECSL Disable Considerations for any Zone

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

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

#### 5.7.7.1 C Code Example to Disable ECSL for C28x Zone1

```
volatile long int *ECSL = (volatile int *)0x5F090; //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++;
// Write the 64-bit password to the CSMKEYx registers
// If this password matches that stored in the
// CSMPWL, then ECSL gets disabled. If the password does not
// match, then the zone remains secure.
// An example password of: // 0x1111222233334444 is used.
*ECSL++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F090
*ECSL++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F092
```

### 5.7.8 Device Unique ID

TI OTP contains a 256-bit value that is made up of both random and sequential parts. This value can be used as a seed for code encryption. The starting address of the value is 0x71140. The first 192 bits are random, the next 32 bits are sequential, and the last 32 bits are a checksum value.

## 5.8 Software

### 5.8.1 DCSM Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
 C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/dcsm

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 5.8.1.1 Empty DCSM Tool Example

FILE: dcsm\_security\_tool.c

This example is an empty project setup for DCSM Tool and Driverlib development. For guidance refer to: [C2000 DCSM Security Tool](#)

## 5.9 DCSM Registers

This section describes the various DCSM Registers.

### 5.9.1 DCSM Base Address Table

**Table 5-4. DCSM Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
DcsmZ1Regs	DCSM_Z1_REGS	DCSM_Z1_BASE	0x0005_F000	YES
DcsmZ2Regs	DCSM_Z2_REGS	DCSM_Z2_BASE	0x0005_F080	YES
DcsmCommonRegs	DCSM_COMMON_REGS	DCSMCOMMON_BASE	0x0005_F0C0	YES
DcsmZ1OtpRegs	DCSM_Z1_OTP	DCSM_Z1OTP_BASE	0x0007_8000	-
DcsmZ2OtpRegs	DCSM_Z2_OTP	DCSM_Z2OTP_BASE	0x0007_8200	-

## 5.9.2 DCSM\_Z1\_REGS Registers

Table 5-5 lists the memory-mapped registers for the DCSM\_Z1\_REGS registers. All register offset addresses not listed in Table 5-5 should be considered as reserved locations and the register contents should not be modified.

**Table 5-5. DCSM\_Z1\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	Z1_LINKPOINTER	Zone 1 Link Pointer		<a href="#">Go</a>
2h	Z1_OTPSECLOCK	Zone 1 OTP Secure Lock		<a href="#">Go</a>
4h	Z1_JLM_ENABLE	Zone 1 JTAGLOCK Enable Register		<a href="#">Go</a>
6h	Z1_LINKPOINTERERR	Link Pointer Error		<a href="#">Go</a>
8h	Z1_GPREG1	Zone 1 General Purpose Register-1		<a href="#">Go</a>
Ah	Z1_GPREG2	Zone 1 General Purpose Register-2		<a href="#">Go</a>
Ch	Z1_GPREG3	Zone 1 General Purpose Register-3		<a href="#">Go</a>
Eh	Z1_GPREG4	Zone 1 General Purpose Register-4		<a href="#">Go</a>
10h	Z1_CSMKEY0	Zone 1 CSM Key 0		<a href="#">Go</a>
12h	Z1_CSMKEY1	Zone 1 CSM Key 1		<a href="#">Go</a>
14h	Z1_CSMKEY2	Zone 1 CSM Key 2		<a href="#">Go</a>
16h	Z1_CSMKEY3	Zone 1 CSM Key 3		<a href="#">Go</a>
18h	Z1_CR	Zone 1 CSM Control Register		<a href="#">Go</a>
1Ah	Z1_GRABSECT1R	Zone 1 Grab Flash Status Register 1		<a href="#">Go</a>
1Ch	Z1_GRABSECT2R	Zone 1 Grab Flash Status Register 2		<a href="#">Go</a>
1Eh	Z1_GRABSECT3R	Zone 1 Grab Flash Status Register 3		<a href="#">Go</a>
20h	Z1_GRABRAM1R	Zone 1 Grab RAM Status Register 1		<a href="#">Go</a>
26h	Z1_EXEONLYSECT1R	Zone 1 Execute Only Flash Status Register 1		<a href="#">Go</a>
28h	Z1_EXEONLYSECT2R	Zone 1 Execute Only Flash Status Register 2		<a href="#">Go</a>
2Ah	Z1_EXEONLYRAM1R	Zone 1 Execute Only RAM Status Register 1		<a href="#">Go</a>
2Eh	Z1_JTAGKEY0	JTAG Unlock Key Register 0		<a href="#">Go</a>
30h	Z1_JTAGKEY1	JTAG Unlock Key Register 1		<a href="#">Go</a>
32h	Z1_JTAGKEY2	JTAG Unlock Key Register 2		<a href="#">Go</a>
34h	Z1_JTAGKEY3	JTAG Unlock Key Register 3		<a href="#">Go</a>
36h	Z1_CMACKKEY0	Secure Boot CMAC Key Status Register 0		<a href="#">Go</a>
38h	Z1_CMACKKEY1	Secure Boot CMAC Key Status Register 1		<a href="#">Go</a>
3Ah	Z1_CMACKKEY2	Secure Boot CMAC Key Status Register 2		<a href="#">Go</a>
3Ch	Z1_CMACKKEY3	Secure Boot CMAC Key Status Register 3		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-6 shows the codes that are used for access types in this section.

**Table 5-6. DCSM\_Z1\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
Reset or Default Value		

**Table 5-6. DCSM\_Z1\_REGS Access Type Codes  
(continued)**

Access Type	Code	Description
<i>-n</i>		Value after reset or the default value

### 5.9.2.1 Z1\_LINKPOINTER Register (Offset = 0h) [Reset = FFFFC000h]

Z1\_LINKPOINTER is shown in [Figure 5-5](#) and described in [Table 5-7](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer

**Figure 5-5. Z1\_LINKPOINTER Register**

31	30	29	28	27	26	25	24	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														LINKPOINTER																	
R-0003FFFFh														R-0h																	

**Table 5-7. Z1\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0003FFFFh	Reserved
13-0	LINKPOINTER	R	0h	This is resolved Link-Pointer value which is generated by looking at the three physical Link-Pointer values loaded from OTP Reset type: SYSRSn

### 5.9.2.2 Z1\_OTPSECLOCK Register (Offset = 2h) [Reset = 00000001h]

Z1\_OTPSECLOCK is shown in [Figure 5-6](#) and described in [Table 5-8](#).

Return to the [Summary Table](#).

Zone 1 OTP Secure Lock

**Figure 5-6. Z1\_OTPSECLOCK 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
PSWDLOCK				RESERVED			JTAGLOCK
R-0h				R-0h			R-1h

**Table 5-8. Z1\_OTPSECLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R	0h	Reserved
7-4	PSWDLOCK	R	0h	Value in this field gets loaded from Z1_PSWDLOCK[3:0] 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. Reset type: XRSn
3-1	RESERVED	R	0h	Reserved
0	JTAGLOCK	R	1h	Reflects the state of the JTAGLOCK feature. 0 : JTAG is not locked 1 : JTAG is locked Reset type: PORESETn

### 5.9.2.3 Z1\_JLM\_ENABLE Register (Offset = 4h) [Reset = 000000Fh]

Z1\_JLM\_ENABLE is shown in [Figure 5-7](#) and described in [Table 5-9](#).

Return to the [Summary Table](#).

Zone 1 JTAGLOCK Enable Register

**Figure 5-7. Z1\_JLM\_ENABLE 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				Z1_JLM_ENABLE			
R-0-0h				R-Fh			

**Table 5-9. Z1\_JLM\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R-0	0h	Reserved
3-0	Z1_JLM_ENABLE	R	Fh	Zone1 JLM_ENABLE register. The value in this field gets loaded from Z1OTP_JLM_ENABLE[3:0] when a read is issued to address location of Z1OTP_JLM_ENABLE in OTP. If Z1OTP_JLM_ENABLE[31:0] is equal to all ones during the load, the JTAGLOCK is not bypassed (is enabled). If the value of Z1OTP_JLM_ENABLE[31:0] is not all ones during the load, the JTAGLOCK is governed as follows by the Z1_JLM_ENABLE bits: 1111 : JTAG/Emulation access is allowed (JTAGLOCK is not enabled) Other values: JTAGLOCK is governed by the JTAGKEY==JTAGPSWD match condition Reset type: PORESETn

### 5.9.2.4 Z1\_LINKPOINTERERR Register (Offset = 6h) [Reset = 0000000h]

Z1\_LINKPOINTERERR is shown in [Figure 5-8](#) and described in [Table 5-10](#).

Return to the [Summary Table](#).

Link Pointer Error

**Figure 5-8. Z1\_LINKPOINTERERR 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		Z1_LINKPOINTERERR													
R-0-0h		R-0h													

**Table 5-10. Z1\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R-0	0h	Reserved
13-0	Z1_LINKPOINTERERR	R	0h	These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP in flash. 0 : No Error. Other : Error on bit positions which is set to 1. Reset type: SYSRSn

### 5.9.2.5 Z1\_GPREG1 Register (Offset = 8h) [Reset = 0000000h]

Z1\_GPREG1 is shown in [Figure 5-9](#) and described in [Table 5-11](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register-1

**Figure 5-9. Z1\_GPREG1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG1																															
R-0h																															

**Table 5-11. Z1\_GPREG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG1	R	0h	This field gets loaded with the contents of Z1OTP_GPREG1 locations in Zone-1-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-1 Reset type: SYSRStn

### 5.9.2.6 Z1\_GPREG2 Register (Offset = Ah) [Reset = 0000000h]

Z1\_GPREG2 is shown in [Figure 5-10](#) and described in [Table 5-12](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register-2

**Figure 5-10. Z1\_GPREG2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG2																															
R-0h																															

**Table 5-12. Z1\_GPREG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG2	R	0h	This field gets loaded with the contents of Z1OTP_GPREG2 locations in Zone-1-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-1 Reset type: SYSRSn

### 5.9.2.7 Z1\_GPREG3 Register (Offset = Ch) [Reset = 0000000h]

Z1\_GPREG3 is shown in [Figure 5-11](#) and described in [Table 5-13](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register-3

**Figure 5-11. Z1\_GPREG3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG3																															
R-0h																															

**Table 5-13. Z1\_GPREG3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG3	R	0h	This field gets loaded with the contents of Z1OTP_GPREG3 locations in Zone-1-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-1 Reset type: SYSRSn

### 5.9.2.8 Z1\_GPREG4 Register (Offset = Eh) [Reset = 0000000h]

Z1\_GPREG4 is shown in [Figure 5-12](#) and described in [Table 5-14](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register-4

**Figure 5-12. Z1\_GPREG4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG4																															
R-0h																															

**Table 5-14. Z1\_GPREG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG4	R	0h	This field gets loaded with the contents of Z1OTP_GPREG4 locations in Zone-1-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-1 Reset type: SYSRSn

### 5.9.2.9 Z1\_CSMKEY0 Register (Offset = 10h) [Reset = 0000000h]

Z1\_CSMKEY0 is shown in [Figure 5-13](#) and described in [Table 5-15](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 0

**Figure 5-13. 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/W-0h																															

**Table 5-15. Z1\_CSMKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY0	R/W	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.) Reset type: SYSRSn

### 5.9.2.10 Z1\_CSMKEY1 Register (Offset = 12h) [Reset = 0000000h]

Z1\_CSMKEY1 is shown in [Figure 5-14](#) and described in [Table 5-16](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 1

**Figure 5-14. 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/W-0h																															

**Table 5-16. Z1\_CSMKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY1	R/W	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.) Reset type: SYSRSn

### 5.9.2.11 Z1\_CSMKEY2 Register (Offset = 14h) [Reset = 0000000h]

Z1\_CSMKEY2 is shown in [Figure 5-15](#) and described in [Table 5-17](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 2

**Figure 5-15. 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/W-0h																															

**Table 5-17. Z1\_CSMKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY2	R/W	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.) Reset type: SYSRStn

### 5.9.2.12 Z1\_CSMKEY3 Register (Offset = 16h) [Reset = 0000000h]

Z1\_CSMKEY3 is shown in [Figure 5-16](#) and described in [Table 5-18](#).

Return to the [Summary Table](#).

Zone 1 CSM Key 3

**Figure 5-16. 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/W-0h																															

**Table 5-18. Z1\_CSMKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1_CSMKEY3	R/W	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.) Reset type: SYSRSn

### 5.9.2.13 Z1\_CR Register (Offset = 18h) [Reset = 00080000h]

Z1\_CR is shown in [Figure 5-17](#) and described in [Table 5-19](#).

Return to the [Summary Table](#).

Zone 1 CSM Control Register

**Figure 5-17. Z1\_CR Register**

31	30	29	28	27	26	25	24
FORCESEC	RESERVED						
R-0/W-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED	ARMED	UNSECURE	ALLONE	ALLZERO	RESERVED		
R-0h	R-0h	R-0h	R-0h	R-1h	R-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

**Table 5-19. Z1\_CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	FORCESEC	R-0/W	0h	A write of '1' to this bit clears the CSMPSWD registers. If writing to this bit after performing an update of the security passwords, it is advised to immediately perform a dummy load of the passwords by initiating a read of the passwords from their flash locations. Reset type: SYSRSn
30-24	RESERVED	R	0h	Reserved
23	RESERVED	R	0h	Reserved
22	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. Reset type: SYSRSn
21	UNSECURE	R	0h	Indicates the state of Zone. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state. Reset type: SYSRSn
20	ALLONE	R	0h	Indicates the state of CSM passwords. 0 : Zone CSM Passwords are not all ones. 1 : Zone CSM Passwords are all ones. Reset type: SYSRSn
19	ALLZERO	R	1h	Indicates the state of CSM passwords. 0 : CSM Passwords are not all zeros. 1 : CSM Passwords are all zero and device is permanently locked. Reset type: SYSRSn
18-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved

**Table 5-19. Z1\_CR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RESERVED	R	0h	Reserved

### 5.9.2.14 Z1\_GRABSECT1R Register (Offset = 1Ah) [Reset = 0000000h]

Z1\_GRABSECT1R is shown in [Figure 5-18](#) and described in [Table 5-20](#).

Return to the [Summary Table](#).

Zone 1 Grab Flash Status Register 1

**Figure 5-18. Z1\_GRABSECT1R Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-20. Z1\_GRABSECT1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-20. Z1\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT11	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT10	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT9	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT8	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT7	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-20. Z1\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT6	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT5	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT4	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT3	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_SECT2	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT1	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-20. Z1\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GRAB_SECT0	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.2.15 Z1\_GRABSECT2R Register (Offset = 1Ch) [Reset = 0000000h]

Z1\_GRABSECT2R is shown in [Figure 5-19](#) and described in [Table 5-21](#).

Return to the [Summary Table](#).

Zone 1 Grab Flash Status Register 2

**Figure 5-19. Z1\_GRABSECT2R Register**

31	30	29	28	27	26	25	24
GRAB_SECT31		GRAB_SECT30		GRAB_SECT29		GRAB_SECT28	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT27		GRAB_SECT26		GRAB_SECT25		GRAB_SECT24	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT23		GRAB_SECT22		GRAB_SECT21		GRAB_SECT20	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT19		GRAB_SECT18		GRAB_SECT17		GRAB_SECT16	
R-0h		R-0h		R-0h		R-0h	

**Table 5-21. Z1\_GRABSECT2R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT31	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT30	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT29	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-21. Z1\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT28	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT27	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT26	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT25	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT24	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT23	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-21. Z1\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT22	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT21	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT20	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT19	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_SECT18	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT17	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-21. Z1\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GRAB_SECT16	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.2.16 Z1\_GRABSECT3R Register (Offset = 1Eh) [Reset = 0000000h]

Z1\_GRABSECT3R is shown in [Figure 5-20](#) and described in [Table 5-22](#).

Return to the [Summary Table](#).

Zone 1 Grab Flash Status Register 3

**Figure 5-20. Z1\_GRABSECT3R Register**

31	30	29	28	27	26	25	24
GRAB_SECT127_120		GRAB_SECT119_112		GRAB_SECT111_104		GRAB_SECT103_96	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT95_88		GRAB_SECT87_80		GRAB_SECT79_72		GRAB_SECT71_64	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT63_56		GRAB_SECT55_48		GRAB_SECT47_40		GRAB_SECT39_32	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		RESERVED		RESERVED	
R-0h		R-0h		R-0h		R-0h	

**Table 5-22. Z1\_GRABSECT3R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT127_120	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT119_112	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT111_104	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-22. Z1\_GRABSECT3R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT103_96	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT95_88	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT87_80	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT79_72	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT71_64	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT63_56	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-22. Z1\_GRABSECT3R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT55_48	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT47_40	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT39_32	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone1. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved
5-4	RESERVED	R	0h	Reserved
3-2	RESERVED	R	0h	Reserved
1-0	RESERVED	R	0h	Reserved

### 5.9.2.17 Z1\_GRABRAM1R Register (Offset = 20h) [Reset = 0000000h]

Z1\_GRABRAM1R is shown in [Figure 5-21](#) and described in [Table 5-23](#).

Return to the [Summary Table](#).

Zone 1 Grab RAM Status Register 1

**Figure 5-21. Z1\_GRABRAM1R Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
GRAB_RAM7		GRAB_RAM6		GRAB_RAM5		GRAB_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_RAM3		GRAB_RAM2		GRAB_RAM1		GRAB_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-23. Z1\_GRABRAM1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-23. Z1\_GRABRAM1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_RAM3	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_RAM2	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_RAM1	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
1-0	GRAB_RAM0	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z1_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone1 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.2.18 Z1\_EXEONLYSECT1R Register (Offset = 26h) [Reset = 0000000h]

Z1\_EXEONLYSECT1R is shown in [Figure 5-22](#) and described in [Table 5-24](#).

Return to the [Summary Table](#).

Zone 1 Execute Only Flash Status Register 1

**Figure 5-22. Z1\_EXEONLYSECT1R Register**

31	30	29	28	27	26	25	24
EXEONLY_SE CT31	EXEONLY_SE CT30	EXEONLY_SE CT29	EXEONLY_SE CT28	EXEONLY_SE CT27	EXEONLY_SE CT26	EXEONLY_SE CT25	EXEONLY_SE CT24
R-0h							
23	22	21	20	19	18	17	16
EXEONLY_SE CT23	EXEONLY_SE CT22	EXEONLY_SE CT21	EXEONLY_SE CT20	EXEONLY_SE CT19	EXEONLY_SE CT18	EXEONLY_SE CT17	EXEONLY_SE CT16
R-0h							
15	14	13	12	11	10	9	8
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h							
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h							

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	EXEONLY_SECT31	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
30	EXEONLY_SECT30	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
29	EXEONLY_SECT29	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	EXEONLY_SECT28	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
27	EXEONLY_SECT27	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
26	EXEONLY_SECT26	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
25	EXEONLY_SECT25	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
24	EXEONLY_SECT24	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
23	EXEONLY_SECT23	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
22	EXEONLY_SECT22	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
21	EXEONLY_SECT21	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
20	EXEONLY_SECT20	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
19	EXEONLY_SECT19	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
18	EXEONLY_SECT18	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
17	EXEONLY_SECT17	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
16	EXEONLY_SECT16	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
15	EXEONLY_SECT15	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	EXEONLY_SECT14	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
12	EXEONLY_SECT12	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
11	EXEONLY_SECT11	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
10	EXEONLY_SECT10	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
9	EXEONLY_SECT9	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
8	EXEONLY_SECT8	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	EXEONLY_SECT7	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
6	EXEONLY_SECT6	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
5	EXEONLY_SECT5	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
4	EXEONLY_SECT4	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
3	EXEONLY_SECT3	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
2	EXEONLY_SECT2	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn
1	EXEONLY_SECT1	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

**Table 5-24. Z1\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone1) 1 : Execute-Only protection is disabled for this sector (only if it is allocated to Zone1) Reset type: SYSRSn

### 5.9.2.19 Z1\_EXEONLYSECT2R Register (Offset = 28h) [Reset = 0000000h]

Z1\_EXEONLYSECT2R is shown in [Figure 5-23](#) and described in [Table 5-25](#).

Return to the [Summary Table](#).

Zone 1 Execute Only Flash Status Register 2

**Figure 5-23. Z1\_EXEONLYSECT2R Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
EXEONLY_SE CT127_120	EXEONLY_SE CT119_112	EXEONLY_SE CT111_104	EXEONLY_SE CT103_96	EXEONLY_SE CT95_88	EXEONLY_SE CT87_80	EXEONLY_SE CT79_72	EXEONLY_SE CT71_64
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 CT63_56	EXEONLY_SE CT55_48	EXEONLY_SE CT47_40	EXEONLY_SE CT39_32	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-25. Z1\_EXEONLYSECT2R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT127_120	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
14	EXEONLY_SECT119_112	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
13	EXEONLY_SECT111_104	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn

**Table 5-25. Z1\_EXEONLYSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT103_96	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
11	EXEONLY_SECT95_88	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
10	EXEONLY_SECT87_80	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
9	EXEONLY_SECT79_72	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
8	EXEONLY_SECT71_64	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
7	EXEONLY_SECT63_56	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
6	EXEONLY_SECT55_48	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn

**Table 5-25. Z1\_EXEONLYSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	EXEONLY_SECT47_40	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
4	EXEONLY_SECT39_32	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone1) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone1) Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

### 5.9.2.20 Z1\_EXEONLYRAM1R Register (Offset = 2Ah) [Reset = 0000000h]

Z1\_EXEONLYRAM1R is shown in [Figure 5-24](#) and described in [Table 5-26](#).

Return to the [Summary Table](#).

Zone 1 Execute Only RAM Status Register 1

**Figure 5-24. Z1\_EXEONLYRAM1R 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							

**Table 5-26. Z1\_EXEONLYRAM1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EXEONLY_RAM7	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
6	EXEONLY_RAM6	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
5	EXEONLY_RAM5	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn

**Table 5-26. Z1\_EXEONLYRAM1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EXEONLY_RAM4	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
3	EXEONLY_RAM3	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
2	EXEONLY_RAM2	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
1	EXEONLY_RAM1	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn
0	EXEONLY_RAM0	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z1_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone1) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone1) Reset type: SYSRSn

### 5.9.2.21 Z1\_JTAGKEY0 Register (Offset = 2Eh) [Reset = 0000000h]

Z1\_JTAGKEY0 is shown in [Figure 5-25](#) and described in [Table 5-27](#).

Return to the [Summary Table](#).

JTAG Unlock Key Register 0

**Figure 5-25. Z1\_JTAGKEY0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY0																															
R-0h																															

**Table 5-27. Z1\_JTAGKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY0	R	0h	Value in this field is scanned in via the JLM scan chain. JTAGKEY is compared to a hidden register that gets dummy loaded when a read is issued to the JTAGPSWD locations in OTP. Reset type: PORESETn

### 5.9.2.22 Z1\_JTAGKEY1 Register (Offset = 30h) [Reset = 00000000h]

Z1\_JTAGKEY1 is shown in [Figure 5-26](#) and described in [Table 5-28](#).

Return to the [Summary Table](#).

JTAG Unlock Key Register 1

**Figure 5-26. Z1\_JTAGKEY1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																KEY1															
																R-0h															

**Table 5-28. Z1\_JTAGKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY1	R	0h	Value in this field is scanned in via the JLM scan chain. JTAGKEY is compared to a hidden register that gets dummy loaded when a read is issued to the JTAGPSWD locations in OTP. Reset type: PORESETn

### 5.9.2.23 Z1\_JTAGKEY2 Register (Offset = 32h) [Reset = 0000000h]

Z1\_JTAGKEY2 is shown in [Figure 5-27](#) and described in [Table 5-29](#).

Return to the [Summary Table](#).

JTAG Unlock Key Register 2

**Figure 5-27. Z1\_JTAGKEY2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY2																															
R-0h																															

**Table 5-29. Z1\_JTAGKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY2	R	0h	Value in this field is scanned in via the JLM scan chain. JTAGKEY is compared to a hidden register that gets dummy loaded when a read is issued to the JTAGPSWD locations in OTP. Reset type: PORESETn

### 5.9.2.24 Z1\_JTAGKEY3 Register (Offset = 34h) [Reset = 0000000h]

Z1\_JTAGKEY3 is shown in [Figure 5-28](#) and described in [Table 5-30](#).

Return to the [Summary Table](#).

JTAG Unlock Key Register 3

**Figure 5-28. Z1\_JTAGKEY3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	KEY3														
																	R-0h														

**Table 5-30. Z1\_JTAGKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY3	R	0h	Value in this field is scanned in via the JLM scan chain. JTAGKEY is compared to a hidden register that gets dummy loaded when a read is issued to the JTAGPSWD locations in OTP. Reset type: PORESETn

### 5.9.2.25 Z1\_CMACEY0 Register (Offset = 36h) [Reset = 0000000h]

Z1\_CMACEY0 is shown in [Figure 5-29](#) and described in [Table 5-31](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key Status Register 0

**Figure 5-29. Z1\_CMACEY0 Register**



**Table 5-31. Z1\_CMACEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY0	R	0h	Value in this field gets loaded from CMACEY0 when a read is issued to its address in OTP. Reset type: SYSRSn

### 5.9.2.26 Z1\_CMACKKEY1 Register (Offset = 38h) [Reset = 0000000h]

Z1\_CMACKKEY1 is shown in [Figure 5-30](#) and described in [Table 5-32](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key Status Register 1

**Figure 5-30. Z1\_CMACKKEY1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	KEY1														
																	R-0h														

**Table 5-32. Z1\_CMACKKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY1	R	0h	Value in this field gets loaded from CMACKKEY1 when a read is issued to its address in OTP. Reset type: SYSRSn

### 5.9.2.27 Z1\_CMACKKEY2 Register (Offset = 3Ah) [Reset = 00000000h]

Z1\_CMACKKEY2 is shown in [Figure 5-31](#) and described in [Table 5-33](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key Status Register 2

**Figure 5-31. Z1\_CMACKKEY2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
																KEY2																					
																R-0h																					

**Table 5-33. Z1\_CMACKKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY2	R	0h	Value in this field gets loaded from CMACKKEY2 when a read is issued to its address in OTP. Reset type: SYSRSn

### 5.9.2.28 Z1\_CMACKKEY3 Register (Offset = 3Ch) [Reset = 00000000h]

Z1\_CMACKKEY3 is shown in [Figure 5-32](#) and described in [Table 5-34](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key Status Register 3

**Figure 5-32. Z1\_CMACKKEY3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	KEY3														
																	R-0h														

**Table 5-34. Z1\_CMACKKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	KEY3	R	0h	Value in this field gets loaded from CMACKKEY3 when a read is issued to its address in OTP. Reset type: SYSRSn

### 5.9.3 DCSM\_Z2\_REGS Registers

Table 5-35 lists the memory-mapped registers for the DCSM\_Z2\_REGS registers. All register offset addresses not listed in Table 5-35 should be considered as reserved locations and the register contents should not be modified.

**Table 5-35. DCSM\_Z2\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	Z2_LINKPOINTER	Zone 2 Link Pointer		<a href="#">Go</a>
2h	Z2_OTPSECLOCK	Zone 2 OTP Secure Lock		<a href="#">Go</a>
6h	Z2_LINKPOINTERERR	Link Pointer Error		<a href="#">Go</a>
8h	Z2_GPREG1	Zone 2 General Purpose Register-1		<a href="#">Go</a>
Ah	Z2_GPREG2	Zone 2 General Purpose Register-2		<a href="#">Go</a>
Ch	Z2_GPREG3	Zone 2 General Purpose Register-3		<a href="#">Go</a>
Eh	Z2_GPREG4	Zone 2 General Purpose Register-4		<a href="#">Go</a>
10h	Z2_CSMKEY0	Zone 2 CSM Key 0		<a href="#">Go</a>
12h	Z2_CSMKEY1	Zone 2 CSM Key 1		<a href="#">Go</a>
14h	Z2_CSMKEY2	Zone 2 CSM Key 2		<a href="#">Go</a>
16h	Z2_CSMKEY3	Zone 2 CSM Key 3		<a href="#">Go</a>
18h	Z2_CR	Zone 2 CSM Control Register		<a href="#">Go</a>
1Ah	Z2_GRABSECT1R	Zone 2 Grab Flash Status Register 1		<a href="#">Go</a>
1Ch	Z2_GRABSECT2R	Zone 2 Grab Flash Status Register 2		<a href="#">Go</a>
1Eh	Z2_GRABSECT3R	Zone 2 Grab Flash Status Register 3		<a href="#">Go</a>
20h	Z2_GRABRAM1R	Zone 2 Grab RAM Status Register 1		<a href="#">Go</a>
26h	Z2_EXEONLYSECT1R	Zone 2 Execute Only Flash Status Register 1		<a href="#">Go</a>
28h	Z2_EXEONLYSECT2R	Zone 2 Execute Only Flash Status Register 2		<a href="#">Go</a>
2Ah	Z2_EXEONLYRAM1R	Zone 2 Execute Only RAM Status Register 1		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-36 shows the codes that are used for access types in this section.

**Table 5-36. DCSM\_Z2\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

### 5.9.3.1 Z2\_LINKPOINTER Register (Offset = 0h) [Reset = FFFFC000h]

Z2\_LINKPOINTER is shown in [Figure 5-33](#) and described in [Table 5-37](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer

**Figure 5-33. Z2\_LINKPOINTER Register**

31	30	29	28	27	26	25	24	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														LINKPOINTER																	
R-0003FFFFh														R-0h																	

**Table 5-37. Z2\_LINKPOINTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0003FFFFh	Reserved
13-0	LINKPOINTER	R	0h	This is resolved Link-Pointer value which is generated by looking at the three physical Link-Pointer values loaded from OTP Reset type: SYSRSn

### 5.9.3.2 Z2\_OTPSECLOCK Register (Offset = 2h) [Reset = 0000001h]

Z2\_OTPSECLOCK is shown in [Figure 5-34](#) and described in [Table 5-38](#).

Return to the [Summary Table](#).

Zone 2 OTP Secure Lock

**Figure 5-34. Z2\_OTPSECLOCK 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
PSWDLOCK				RESERVED			JTAGLOCK
R-0h				R-0h			R-1h

**Table 5-38. Z2\_OTPSECLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R	0h	Reserved
7-4	PSWDLOCK	R	0h	Value in this field gets loaded from Z2_PSWDLOCK[3:0] when a read is issued to address location of 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. Reset type: XRSn
3-1	RESERVED	R	0h	Reserved
0	JTAGLOCK	R	1h	Reflects the state of the JTAGLOCK feature. 0 : JTAG is not locked 1 : JTAG is locked This bit is a copy of the Z1_OTPSECLOCK.JTAGLOCK bit. Reset type: PORESETn

### 5.9.3.3 Z2\_LINKPOINTERERR Register (Offset = 6h) [Reset = 0000000h]

Z2\_LINKPOINTERERR is shown in [Figure 5-35](#) and described in [Table 5-39](#).

Return to the [Summary Table](#).

Link Pointer Error

**Figure 5-35. Z2\_LINKPOINTERERR 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		Z2_LINKPOINTERERR													
R-0-0h		R-0h													

**Table 5-39. Z2\_LINKPOINTERERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R-0	0h	Reserved
13-0	Z2_LINKPOINTERERR	R	0h	These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded of OTP in flash. 0 : No Error. Other : Error on bit positions which is set to 1. Reset type: SYSRSn

### 5.9.3.4 Z2\_GPREG1 Register (Offset = 8h) [Reset = 0000000h]

Z2\_GPREG1 is shown in [Figure 5-36](#) and described in [Table 5-40](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register-1

**Figure 5-36. Z2\_GPREG1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG1																															
R-0h																															

**Table 5-40. Z2\_GPREG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG1	R	0h	This field gets loaded with the contents of Z2OTP_GPREG1 locations in Zone-2-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-2 Reset type: SYSRSn

### 5.9.3.5 Z2\_GPREG2 Register (Offset = Ah) [Reset = 0000000h]

Z2\_GPREG2 is shown in [Figure 5-37](#) and described in [Table 5-41](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register-2

**Figure 5-37. Z2\_GPREG2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG2																															
R-0h																															

**Table 5-41. Z2\_GPREG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG2	R	0h	This field gets loaded with the contents of Z2OTP_GPREG2 locations in Zone-2-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-2 Reset type: SYSRSn

### 5.9.3.6 Z2\_GPREG3 Register (Offset = Ch) [Reset = 0000000h]

Z2\_GPREG3 is shown in [Figure 5-38](#) and described in [Table 5-42](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register-3

**Figure 5-38. Z2\_GPREG3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG3																															
R-0h																															

**Table 5-42. Z2\_GPREG3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG3	R	0h	This field gets loaded with the contents of Z2OTP_GPREG3 locations in Zone-2-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-2 Reset type: SYSRSn

### 5.9.3.7 Z2\_GPREG4 Register (Offset = Eh) [Reset = 0000000h]

Z2\_GPREG4 is shown in [Figure 5-39](#) and described in [Table 5-43](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register-4

**Figure 5-39. Z2\_GPREG4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPREG4																															
R-0h																															

**Table 5-43. Z2\_GPREG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	GPREG4	R	0h	This field gets loaded with the contents of Z2OTP_GPREG4 locations in Zone-2-USER-OTP when a dummy read is issued to that address. Users can use this register to load any general purpose non-volatile content from USER-OTP of Zone-2 Reset type: SYSRSn

### 5.9.3.8 Z2\_CSMKEY0 Register (Offset = 10h) [Reset = 0000000h]

Z2\_CSMKEY0 is shown in [Figure 5-40](#) and described in [Table 5-44](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 0

**Figure 5-40. 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/W-0h																															

**Table 5-44. Z2\_CSMKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY0	R/W	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.) Reset type: SYSRSn

### 5.9.3.9 Z2\_CSMKEY1 Register (Offset = 12h) [Reset = 0000000h]

Z2\_CSMKEY1 is shown in [Figure 5-41](#) and described in [Table 5-45](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 1

**Figure 5-41. 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/W-0h																															

**Table 5-45. Z2\_CSMKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY1	R/W	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.) Reset type: SYSRStn

### 5.9.3.10 Z2\_CSMKEY2 Register (Offset = 14h) [Reset = 0000000h]

Z2\_CSMKEY2 is shown in [Figure 5-42](#) and described in [Table 5-46](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 2

**Figure 5-42. 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/W-0h																															

**Table 5-46. Z2\_CSMKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY2	R/W	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.) Reset type: SYSRSn

### 5.9.3.11 Z2\_CSMKEY3 Register (Offset = 16h) [Reset = 0000000h]

Z2\_CSMKEY3 is shown in [Figure 5-43](#) and described in [Table 5-47](#).

Return to the [Summary Table](#).

Zone 2 CSM Key 3

**Figure 5-43. 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/W-0h																															

**Table 5-47. Z2\_CSMKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2_CSMKEY3	R/W	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.) Reset type: SYSRSn

### 5.9.3.12 Z2\_CR Register (Offset = 18h) [Reset = 00080000h]

Z2\_CR is shown in [Figure 5-44](#) and described in [Table 5-48](#).

Return to the [Summary Table](#).

Zone 2 CSM Control Register

**Figure 5-44. Z2\_CR Register**

31	30	29	28	27	26	25	24
FORCESEC	RESERVED						
R-0/W-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED	ARMED	UNSECURE	ALLONE	ALLZERO	RESERVED		
R-0h	R-0h	R-0h	R-0h	R-1h	R-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

**Table 5-48. Z2\_CR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	FORCESEC	R-0/W	0h	A write of '1' to this bit clears the CSMPSWD registers. If writing to this bit after performing an update of the security passwords, it is advised to immediately perform a dummy load of the passwords by initiating a read of the passwords from their flash locations. Reset type: SYSRSn
30-24	RESERVED	R	0h	Reserved
23	RESERVED	R	0h	Reserved
22	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. Reset type: SYSRSn
21	UNSECURE	R	0h	Indicates the state of Zone. 0 : Zone is in lock(secure) state. 1 : Zone is in unlock(unsecure) state. Reset type: SYSRSn
20	ALLONE	R	0h	Indicates the state of CSM passwords. 0 : Zone CSM Passwords are not all ones. 1 : Zone CSM Passwords are all ones. Reset type: SYSRSn
19	ALLZERO	R	1h	Indicates the state of CSM passwords. 0 : CSM Passwords are not all zeros. 1 : CSM Passwords are all zero and device is permanently locked. Reset type: SYSRSn
18-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R-0	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved

**Table 5-48. Z2\_CR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RESERVED	R	0h	Reserved

### 5.9.3.13 Z2\_GRABSECT1R Register (Offset = 1Ah) [Reset = 0000000h]

Z2\_GRABSECT1R is shown in [Figure 5-45](#) and described in [Table 5-49](#).

Return to the [Summary Table](#).

Zone 2 Grab Flash Status Register 1

**Figure 5-45. Z2\_GRABSECT1R Register**

31	30	29	28	27	26	25	24
GRAB_SECT15		GRAB_SECT14		GRAB_SECT13		GRAB_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT11		GRAB_SECT10		GRAB_SECT9		GRAB_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT7		GRAB_SECT6		GRAB_SECT5		GRAB_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_SECT3		GRAB_SECT2		GRAB_SECT1		GRAB_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-49. Z2\_GRABSECT1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT15	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT14	R	0h	Value in this field gets loaded from Z2_GRABSECT1[29:28] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 14 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 14 to Zone2. 10 : No request for Bank 0 Flash Sector 14 11 : No request for Bank 0 Flash Sector 14 when this zone is UNLOCKED. Else Bank 0 Flash Sector 14 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT13	R	0h	Value in this field gets loaded from Z2_GRABSECT1[27:26] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 13 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 13 to Zone2. 10 : No request for Bank 0 Flash Sector 13 11 : No request for Bank 0 Flash Sector 13 when this zone is UNLOCKED. Else Bank 0 Flash Sector 13 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-49. Z2\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT12	R	0h	Value in this field gets loaded from Z2_GRABSECT1[25:24] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 12 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 12 to Zone2. 10 : No request for Bank 0 Flash Sector 12 11 : No request for Bank 0 Flash Sector 12 when this zone is UNLOCKED. Else Bank 0 Flash Sector 12 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT11	R	0h	Value in this field gets loaded from Z2_GRABSECT1[23:22] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 11 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 11 to Zone2. 10 : No request for Bank 0 Flash Sector 11 11 : No request for Bank 0 Flash Sector 11 when this zone is UNLOCKED. Else Bank 0 Flash Sector 11 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT10	R	0h	Value in this field gets loaded from Z2_GRABSECT1[21:20] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 10 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 10 to Zone2. 10 : No request for Bank 0 Flash Sector 10 11 : No request for Bank 0 Flash Sector 10 when this zone is UNLOCKED. Else Bank 0 Flash Sector 10 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT9	R	0h	Value in this field gets loaded from Z2_GRABSECT1[19:18] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 9 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 9 to Zone2. 10 : No request for Bank 0 Flash Sector 9 11 : No request for Bank 0 Flash Sector 9 when this zone is UNLOCKED. Else Bank 0 Flash Sector 9 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT8	R	0h	Value in this field gets loaded from Z2_GRABSECT1[17:16] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 8 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 8 to Zone2. 10 : No request for Bank 0 Flash Sector 8 11 : No request for Bank 0 Flash Sector 8 when this zone is UNLOCKED. Else Bank 0 Flash Sector 8 is inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT7	R	0h	Value in this field gets loaded from Z2_GRABSECT1[15:14] when a read is issued to address location of Z2_GRABSECT1 in OTP. 00 : Invalid. Bank 0 Flash Sector 7 is inaccessible. 01 : Request to allocate Bank 0 Flash Sector 7 to Zone2. 10 : No request for Bank 0 Flash Sector 7 11 : No request for Bank 0 Flash Sector 7 when this zone is UNLOCKED. Else Bank 0 Flash Sector 7 is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-49. Z2\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT6	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT5	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT4	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT3	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_SECT2	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT1	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-49. Z2\_GRABSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GRAB_SECT0	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT1 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.3.14 Z2\_GRABSECT2R Register (Offset = 1Ch) [Reset = 0000000h]

Z2\_GRABSECT2R is shown in [Figure 5-46](#) and described in [Table 5-50](#).

Return to the [Summary Table](#).

Zone 2 Grab Flash Status Register 2

**Figure 5-46. Z2\_GRABSECT2R Register**

31	30	29	28	27	26	25	24
GRAB_SECT31	GRAB_SECT30		GRAB_SECT29		GRAB_SECT28		
R-0h	R-0h		R-0h		R-0h		
23	22	21	20	19	18	17	16
GRAB_SECT27	GRAB_SECT26		GRAB_SECT25		GRAB_SECT24		
R-0h	R-0h		R-0h		R-0h		
15	14	13	12	11	10	9	8
GRAB_SECT23	GRAB_SECT22		GRAB_SECT21		GRAB_SECT20		
R-0h	R-0h		R-0h		R-0h		
7	6	5	4	3	2	1	0
GRAB_SECT19	GRAB_SECT18		GRAB_SECT17		GRAB_SECT16		
R-0h	R-0h		R-0h		R-0h		

**Table 5-50. Z2\_GRABSECT2R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT31	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT30	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT29	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-50. Z2\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT28	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT27	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT26	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT25	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT24	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT23	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-50. Z2\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT22	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT21	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT20	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_SECT19	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_SECT18	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_SECT17	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-50. Z2\_GRABSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	GRAB_SECT16	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT2 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.3.15 Z2\_GRABSECT3R Register (Offset = 1Eh) [Reset = 0000000h]

Z2\_GRABSECT3R is shown in [Figure 5-47](#) and described in [Table 5-51](#).

Return to the [Summary Table](#).

Zone 2 Grab Flash Status Register 3

**Figure 5-47. Z2\_GRABSECT3R Register**

31	30	29	28	27	26	25	24
GRAB_SECT127_120		GRAB_SECT119_112		GRAB_SECT111_104		GRAB_SECT103_96	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
GRAB_SECT95_88		GRAB_SECT87_80		GRAB_SECT79_72		GRAB_SECT71_64	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
GRAB_SECT63_56		GRAB_SECT55_48		GRAB_SECT47_40		GRAB_SECT39_32	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		RESERVED		RESERVED	
R-0h		R-0h		R-0h		R-0h	

**Table 5-51. Z2\_GRABSECT3R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GRAB_SECT127_120	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
29-28	GRAB_SECT119_112	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
27-26	GRAB_SECT111_104	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-51. Z2\_GRABSECT3R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
25-24	GRAB_SECT103_96	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
23-22	GRAB_SECT95_88	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
21-20	GRAB_SECT87_80	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
19-18	GRAB_SECT79_72	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
17-16	GRAB_SECT71_64	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
15-14	GRAB_SECT63_56	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-51. Z2\_GRABSECT3R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GRAB_SECT55_48	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_SECT47_40	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
9-8	GRAB_SECT39_32	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABSECT3 in the SECURITY sector. 00 : Invalid. Sectors are inaccessible. 01 : Request to allocate these sectors to Zone2. 10 : No request for these sectors. 11 : No request for these sectors when this zone is UNLOCKED. Else these sectors are inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved
5-4	RESERVED	R	0h	Reserved
3-2	RESERVED	R	0h	Reserved
1-0	RESERVED	R	0h	Reserved

### 5.9.3.16 Z2\_GRABRAM1R Register (Offset = 20h) [Reset = 0000000h]

Z2\_GRABRAM1R is shown in [Figure 5-48](#) and described in [Table 5-52](#).

Return to the [Summary Table](#).

Zone 2 Grab RAM Status Register 1

**Figure 5-48. Z2\_GRABRAM1R Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
GRAB_RAM7		GRAB_RAM6		GRAB_RAM5		GRAB_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
GRAB_RAM3		GRAB_RAM2		GRAB_RAM1		GRAB_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-52. Z2\_GRABRAM1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	GRAB_RAM7	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
13-12	GRAB_RAM6	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
11-10	GRAB_RAM5	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

**Table 5-52. Z2\_GRABRAM1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	GRAB_RAM4	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
7-6	GRAB_RAM3	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
5-4	GRAB_RAM2	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
3-2	GRAB_RAM1	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn
1-0	GRAB_RAM0	R	0h	Value in this field gets loaded from the equivalent bit field when a read is issued to the active ZSB address location of Z2_GRABRAM1 in the SECURITY sector. 00 : Invalid. This section of RAM is inaccessible. 01 : Request to allocate this section of RAM to Zone2 10 : No request for this section of RAM 11 : No request for this section of RAM when this zone is UNLOCKED. This section of RAM is inaccessible if this zone is LOCKED. Reset type: SYSRSn

### 5.9.3.17 Z2\_EXEONLYSECT1R Register (Offset = 26h) [Reset = 0000000h]

Z2\_EXEONLYSECT1R is shown in [Figure 5-49](#) and described in [Table 5-53](#).

Return to the [Summary Table](#).

Zone 2 Execute Only Flash Status Register 1

**Figure 5-49. Z2\_EXEONLYSECT1R Register**

31	30	29	28	27	26	25	24
EXEONLY_SE CT31	EXEONLY_SE CT30	EXEONLY_SE CT29	EXEONLY_SE CT28	EXEONLY_SE CT27	EXEONLY_SE CT26	EXEONLY_SE CT25	EXEONLY_SE CT24
R-0h							
23	22	21	20	19	18	17	16
EXEONLY_SE CT23	EXEONLY_SE CT22	EXEONLY_SE CT21	EXEONLY_SE CT20	EXEONLY_SE CT19	EXEONLY_SE CT18	EXEONLY_SE CT17	EXEONLY_SE CT16
R-0h							
15	14	13	12	11	10	9	8
EXEONLY_SE CT15	EXEONLY_SE CT14	EXEONLY_SE CT13	EXEONLY_SE CT12	EXEONLY_SE CT11	EXEONLY_SE CT10	EXEONLY_SE CT9	EXEONLY_SE CT8
R-0h							
7	6	5	4	3	2	1	0
EXEONLY_SE CT7	EXEONLY_SE CT6	EXEONLY_SE CT5	EXEONLY_SE CT4	EXEONLY_SE CT3	EXEONLY_SE CT2	EXEONLY_SE CT1	EXEONLY_SE CT0
R-0h							

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	EXEONLY_SECT31	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
30	EXEONLY_SECT30	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
29	EXEONLY_SECT29	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	EXEONLY_SECT28	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
27	EXEONLY_SECT27	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
26	EXEONLY_SECT26	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
25	EXEONLY_SECT25	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
24	EXEONLY_SECT24	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
23	EXEONLY_SECT23	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
22	EXEONLY_SECT22	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
21	EXEONLY_SECT21	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
20	EXEONLY_SECT20	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
19	EXEONLY_SECT19	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
18	EXEONLY_SECT18	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
17	EXEONLY_SECT17	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
16	EXEONLY_SECT16	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
15	EXEONLY_SECT15	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	EXEONLY_SECT14	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
13	EXEONLY_SECT13	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
12	EXEONLY_SECT12	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
11	EXEONLY_SECT11	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
10	EXEONLY_SECT10	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
9	EXEONLY_SECT9	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
8	EXEONLY_SECT8	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	EXEONLY_SECT7	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
6	EXEONLY_SECT6	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
5	EXEONLY_SECT5	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
4	EXEONLY_SECT4	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
3	EXEONLY_SECT3	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
2	EXEONLY_SECT2	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn
1	EXEONLY_SECT1	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

**Table 5-53. Z2\_EXEONLYSECT1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EXEONLY_SECT0	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this sector (only if it is allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if it is allocated to Zone2) Reset type: SYSRSn

### 5.9.3.18 Z2\_EXEONLYSECT2R Register (Offset = 28h) [Reset = 0000000h]

Z2\_EXEONLYSECT2R is shown in [Figure 5-50](#) and described in [Table 5-54](#).

Return to the [Summary Table](#).

Zone 2 Execute Only Flash Status Register 2

**Figure 5-50. Z2\_EXEONLYSECT2R Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
EXEONLY_SE CT127_120	EXEONLY_SE CT119_112	EXEONLY_SE CT111_104	EXEONLY_SE CT103_96	EXEONLY_SE CT95_88	EXEONLY_SE CT87_80	EXEONLY_SE CT79_72	EXEONLY_SE CT71_64
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 CT63_56	EXEONLY_SE CT55_48	EXEONLY_SE CT47_40	EXEONLY_SE CT39_32	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-54. Z2\_EXEONLYSECT2R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EXEONLY_SECT127_120	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
14	EXEONLY_SECT119_112	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
13	EXEONLY_SECT111_104	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn

**Table 5-54. Z2\_EXEONLYSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	EXEONLY_SECT103_96	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
11	EXEONLY_SECT95_88	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
10	EXEONLY_SECT87_80	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
9	EXEONLY_SECT79_72	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
8	EXEONLY_SECT71_64	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
7	EXEONLY_SECT63_56	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
6	EXEONLY_SECT55_48	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn

**Table 5-54. Z2\_EXEONLYSECT2R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	EXEONLY_SECT47_40	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
4	EXEONLY_SECT39_32	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYSECT2 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for 1527these sectors (only if they are allocated to Zone2) 1 : Execute-Only protection is disabled for these sectors (only if they are allocated to Zone2) Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

### 5.9.3.19 Z2\_EXEONLYRAM1R Register (Offset = 2Ah) [Reset = 0000000h]

Z2\_EXEONLYRAM1R is shown in [Figure 5-51](#) and described in [Table 5-55](#).

Return to the [Summary Table](#).

Zone 2 Execute Only RAM Status Register 1

**Figure 5-51. Z2\_EXEONLYRAM1R 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							

**Table 5-55. Z2\_EXEONLYRAM1R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	EXEONLY_RAM7	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
6	EXEONLY_RAM6	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
5	EXEONLY_RAM5	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn

**Table 5-55. Z2\_EXEONLYRAM1R Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EXEONLY_RAM4	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
3	EXEONLY_RAM3	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
2	EXEONLY_RAM2	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
1	EXEONLY_RAM1	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn
0	EXEONLY_RAM0	R	0h	Value in this bit gets loaded from the equivalent bit when a read is issued to the Z2_EXEONLYRAM1 address location in the SECURITY sector. 0 : Execute-Only protection is enabled for this section of RAM (only if it's allocated to Zone2) 1 : Execute-Only protection is disabled for this section of RAM (only if it's allocated to Zone2) Reset type: SYSRSn

### 5.9.4 DCSM\_COMMON\_REGS Registers

Table 5-56 lists the memory-mapped registers for the DCSM\_COMMON\_REGS registers. All register offset addresses not listed in Table 5-56 should be considered as reserved locations and the register contents should not be modified.

**Table 5-56. DCSM\_COMMON\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	FLSEM	Flash Wrapper Semaphore Register	EALLOW	<a href="#">Go</a>
8h	SECTSTAT1	Flash Sectors Status Register 1		<a href="#">Go</a>
Ah	SECTSTAT2	Flash Sectors Status Register 2		<a href="#">Go</a>
Ch	SECTSTAT3	Flash Sectors Status Register 3		<a href="#">Go</a>
10h	RAMSTAT1	RAM Status Register 1		<a href="#">Go</a>
18h	SECERRSTAT	Security Error Status Register		<a href="#">Go</a>
1Ah	SECERRCLR	Security Error Clear Register		<a href="#">Go</a>
1Ch	SECERRFRC	Security Error Force Register		<a href="#">Go</a>
1Eh	DENYCODE	Flash Authorization Denial Code		<a href="#">Go</a>
20h	RAMOPENSTAT	RAM Security Open Status Register		<a href="#">Go</a>
22h	RAMOPENFRC	RAM Security Open Force Register	EALLOW	<a href="#">Go</a>
24h	RAMOPENCLR	RAM Security Open Clear Register	EALLOW	<a href="#">Go</a>
26h	RAMOPENLOCK	RAMOPEN Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-57 shows the codes that are used for access types in this section.

**Table 5-57. DCSM\_COMMON\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value

### 5.9.4.1 FLSEM Register (Offset = 0h) [Reset = 0000000h]

FLSEM is shown in [Figure 5-52](#) and described in [Table 5-58](#).

Return to the [Summary Table](#).

Flash Wrapper Semaphore Register

**Figure 5-52. FLSEM Register**

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

**Table 5-58. FLSEM Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-8	KEY	R-0/W	0h	Writing a value 0xA5 into this field will allow the writing of the SEM bits, else writes are ignored. Reads will return 0. Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1-0	SEM	R/W	0h	00 : Flash Wrapper registers can be written by code running from anywhere without any restriction. 01 : Flash Wrapper registers can be written by code running from Zone1 security zone. 10 : Flash Wrapper registers can be written by code running from Zone2 security zone 11 : Flash Wrapper registers can be written by code running from anywhere without any restriction Allowed State Transitions in this field. 00 TO 11 : Not allowed. 11 TO 00 : Not allowed. 00/11 TO 01 : Code running from Zone1 only can perform this transition. 01 TO 00/11 : Code running from Zone1 only can perform this transition. 00/11 TO 10 : Code running from Zone2 only can perform this transition. 10 TO 00/11 : Code running from Zone2 can perform this transition 10 TO 01 : Not allowed. 01 TO 10 : Not allowed. Reset type: SYSRSn

### 5.9.4.2 SECTSTAT1 Register (Offset = 8h) [Reset = 0000000h]

SECTSTAT1 is shown in [Figure 5-53](#) and described in [Table 5-59](#).

Return to the [Summary Table](#).

Flash Sectors Status Register 1

**Figure 5-53. SECTSTAT1 Register**

31	30	29	28	27	26	25	24
STATUS_SECT15		STATUS_SECT14		STATUS_SECT13		STATUS_SECT12	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECT11		STATUS_SECT10		STATUS_SECT9		STATUS_SECT8	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECT7		STATUS_SECT6		STATUS_SECT5		STATUS_SECT4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_SECT3		STATUS_SECT2		STATUS_SECT1		STATUS_SECT0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-59. SECTSTAT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	STATUS_SECT15	R	0h	Reflects the status of flash sector 15. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
29-28	STATUS_SECT14	R	0h	Reflects the status of flash sector 14. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
27-26	STATUS_SECT13	R	0h	Reflects the status of flash sector 13. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
25-24	STATUS_SECT12	R	0h	Reflects the status of flash sector 12. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-59. SECTSTAT1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	STATUS_SECT11	R	0h	Reflects the status of flash sector 11. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
21-20	STATUS_SECT10	R	0h	Reflects the status of flash sector 10. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
19-18	STATUS_SECT9	R	0h	Reflects the status of flash sector 9. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
17-16	STATUS_SECT8	R	0h	Reflects the status of flash sector 8. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
15-14	STATUS_SECT7	R	0h	Reflects the status of flash sector 7. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_SECT6	R	0h	Reflects the status of flash sector 6. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_SECT5	R	0h	Reflects the status of flash sector 5. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
9-8	STATUS_SECT4	R	0h	Reflects the status of flash sector 4. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-59. SECTSTAT1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	STATUS_SECT3	R	0h	Reflects the status of flash sector 3. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_SECT2	R	0h	Reflects the status of flash sector 2. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_SECT1	R	0h	Reflects the status of flash sector 1. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_SECT0	R	0h	Reflects the status of flash sector 0. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

### 5.9.4.3 SECTSTAT2 Register (Offset = Ah) [Reset = 0000000h]

SECTSTAT2 is shown in [Figure 5-54](#) and described in [Table 5-60](#).

Return to the [Summary Table](#).

Flash Sectors Status Register 2

**Figure 5-54. SECTSTAT2 Register**

31	30	29	28	27	26	25	24
STATUS_SECT31		STATUS_SECT30		STATUS_SECT29		STATUS_SECT28	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECT27		STATUS_SECT26		STATUS_SECT25		STATUS_SECT24	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECT23		STATUS_SECT22		STATUS_SECT21		STATUS_SECT20	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_SECT19		STATUS_SECT18		STATUS_SECT17		STATUS_SECT16	
R-0h		R-0h		R-0h		R-0h	

**Table 5-60. SECTSTAT2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	STATUS_SECT31	R	0h	Reflects the status of flash sector 31. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
29-28	STATUS_SECT30	R	0h	Reflects the status of flash sector 30. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
27-26	STATUS_SECT29	R	0h	Reflects the status of flash sector 29. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
25-24	STATUS_SECT28	R	0h	Reflects the status of flash sector 28. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-60. SECTSTAT2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	STATUS_SECT27	R	0h	Reflects the status of flash sector 27. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
21-20	STATUS_SECT26	R	0h	Reflects the status of flash sector 26. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
19-18	STATUS_SECT25	R	0h	Reflects the status of flash sector 25. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
17-16	STATUS_SECT24	R	0h	Reflects the status of flash sector 24. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
15-14	STATUS_SECT23	R	0h	Reflects the status of flash sector 23. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_SECT22	R	0h	Reflects the status of flash sector 22. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_SECT21	R	0h	Reflects the status of flash sector 21. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
9-8	STATUS_SECT20	R	0h	Reflects the status of flash sector 20. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-60. SECTSTAT2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	STATUS_SECT19	R	0h	Reflects the status of flash sector 19. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_SECT18	R	0h	Reflects the status of flash sector 18. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_SECT17	R	0h	Reflects the status of flash sector 17. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_SECT16	R	0h	Reflects the status of flash sector 16. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

#### 5.9.4.4 SECTSTAT3 Register (Offset = Ch) [Reset = 0000000h]

SECTSTAT3 is shown in [Figure 5-55](#) and described in [Table 5-61](#).

Return to the [Summary Table](#).

Flash Sectors Status Register 3

**Figure 5-55. SECTSTAT3 Register**

31	30	29	28	27	26	25	24
STATUS_SECT127_120		STATUS_SECT119_112		STATUS_SECT111_104		STATUS_SECT103_96	
R-0h		R-0h		R-0h		R-0h	
23	22	21	20	19	18	17	16
STATUS_SECT95_88		STATUS_SECT87_80		STATUS_SECT79_72		STATUS_SECT71_64	
R-0h		R-0h		R-0h		R-0h	
15	14	13	12	11	10	9	8
STATUS_SECT63_56		STATUS_SECT55_48		STATUS_SECT47_40		STATUS_SECT39_32	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
RESERVED		RESERVED		RESERVED		RESERVED	
R-0-0h		R-0-0h		R-0-0h		R-0-0h	

**Table 5-61. SECTSTAT3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	STATUS_SECT127_120	R	0h	Reflects the status of flash sectors 127-120. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
29-28	STATUS_SECT119_112	R	0h	Reflects the status of flash sectors 119-112. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
27-26	STATUS_SECT111_104	R	0h	Reflects the status of flash sectors 111-104. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
25-24	STATUS_SECT103_96	R	0h	Reflects the status of flash sectors 103-96. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-61. SECTSTAT3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23-22	STATUS_SECT95_88	R	0h	Reflects the status of flash sectors 95-88. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
21-20	STATUS_SECT87_80	R	0h	Reflects the status of flash sectors 87-80. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
19-18	STATUS_SECT79_72	R	0h	Reflects the status of flash sectors 79-72. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
17-16	STATUS_SECT71_64	R	0h	Reflects the status of flash sectors 71-64. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
15-14	STATUS_SECT63_56	R	0h	Reflects the status of flash sectors 63-56. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_SECT55_48	R	0h	Reflects the status of flash sectors 55-48. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_SECT47_40	R	0h	Reflects the status of flash sectors 47-40. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
9-8	STATUS_SECT39_32	R	0h	Reflects the status of flash sectors 39-32. 00 : Sector is in-accessible 01 : Sector belongs to Zone1. 10 : Sector belongs to Zone2. 11: Sector is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
7-6	RESERVED	R-0	0h	Reserved
5-4	RESERVED	R-0	0h	Reserved
3-2	RESERVED	R-0	0h	Reserved

**Table 5-61. SECTSTAT3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	RESERVED	R-0	0h	Reserved

### 5.9.4.5 RAMSTAT1 Register (Offset = 10h) [Reset = 0000000h]

RAMSTAT1 is shown in [Figure 5-56](#) and described in [Table 5-62](#).

Return to the [Summary Table](#).

RAM Status Register 1

**Figure 5-56. RAMSTAT1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
STATUS_RAM7		STATUS_RAM6		STATUS_RAM5		STATUS_RAM4	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
STATUS_RAM3		STATUS_RAM2		STATUS_RAM1		STATUS_RAM0	
R-0h		R-0h		R-0h		R-0h	

**Table 5-62. RAMSTAT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-14	STATUS_RAM7	R	0h	Reflects the status Section D of LS1 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
13-12	STATUS_RAM6	R	0h	Reflects the status Section C of LS1 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
11-10	STATUS_RAM5	R	0h	Reflects the status Section B of LS1 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
9-8	STATUS_RAM4	R	0h	Reflects the status Section A of LS1 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

**Table 5-62. RAMSTAT1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	STATUS_RAM3	R	0h	Reflects the status Section D of LS0 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
5-4	STATUS_RAM2	R	0h	Reflects the status Section C of LS0 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
3-2	STATUS_RAM1	R	0h	Reflects the status Section B of LS0 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn
1-0	STATUS_RAM0	R	0h	Reflects the status Section A of LS0 RAM. 00 : RAM is in-accessible 01 : RAM belongs to Zone1. 10 : RAM belongs to Zone2. 11: RAM is un-secure and code running in both zone have full access to it. Reset type: SYSRSn

### 5.9.4.6 SECERRSTAT Register (Offset = 18h) [Reset = 0000000h]

SECERRSTAT is shown in [Figure 5-57](#) and described in [Table 5-63](#).

Return to the [Summary Table](#).

Security Error Status Register

**Figure 5-57. SECERRSTAT Register**

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

**Table 5-63. SECERRSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	ERR	R	0h	This bit indicates if any error has occurred in the load of any security configuration from USER-OTP. 0: No error has occurred in the load of security information from USER-OTP 1: Error has occurred in the load of security information from USER-OTP Reset type: PORESETn

### 5.9.4.7 SECERRCLR Register (Offset = 1Ah) [Reset = 0000000h]

SECERRCLR is shown in [Figure 5-58](#) and described in [Table 5-64](#).

Return to the [Summary Table](#).

Security Error Clear Register

**Figure 5-58. SECERRCLR Register**

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

**Table 5-64. SECERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	ERR	R-0/W1S	0h	A write of '1' clears the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'. Reset type: N/A

### 5.9.4.8 SECERRFRC Register (Offset = 1Ch) [Reset = 0000000h]

SECERRFRC is shown in [Figure 5-59](#) and described in [Table 5-65](#).

Return to the [Summary Table](#).

Security Error Force Register

**Figure 5-59. SECERRFRC 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															ERR
R-0-0h															R-0/ W1S-0 h

**Table 5-65. SECERRFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	In order to write to the ERR 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 write to ERR. Reads will return 0. Reset type: N/A
15-1	RESERVED	R-0	0h	Reserved
0	ERR	R-0/W1S	0h	A write of '1', along with the proper KEY, sets the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'. Reset type: N/A

### 5.9.4.9 DENYCODE Register (Offset = 1Eh) [Reset = 0000000h]

DENYCODE is shown in [Figure 5-60](#) and described in [Table 5-66](#).

Return to the [Summary Table](#).

Flash Authorization Denial Code

**Figure 5-60. DENYCODE 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
ILLSIZE	ILLCMD	ILLMODECH	ILLRDVER	ILLERASE	ILLPROG	ILLADDR	BLOCKED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 5-66. DENYCODE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R-0	0h	Reserved
7	ILLSIZE	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because an illegal command size was requested. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal command size 1 : Flash operation was stopped due to an illegal command size Reset type: SYSRSn
6	ILLCMD	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because an illegal command type was requested. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal command type 1 : Flash operation was stopped due to an illegal command type Reset type: SYSRSn
5	ILLMODECH	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because a mode change command tried to move it into a reserved mode. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal mode change 1 : Flash operation was stopped due to an illegal mode change Reset type: SYSRSn
4	ILLRDVER	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because a read verify command provided an illegal address. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal read verify address 1 : Flash operation was stopped due to an illegal read verify address Reset type: SYSRSn

**Table 5-66. DENYCODE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	ILLERASE	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because an erase command provided an illegal address. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal erase address 1 : Flash operation was stopped due to an illegal erase address Reset type: SYSRSn
2	ILLPROG	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because a programming command provided an illegal address. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to an illegal programming address 1 : Flash operation was stopped due to an illegal programming address Reset type: SYSRSn
1	ILLADDR	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because the command provided contained a non-flash address. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to a non-flash address 1 : Flash operation was stopped due to a non-flash address Reset type: SYSRSn
0	BLOCKED	R	0h	This bit indicates the DCSM stopped a Flash Controller operation because the DCSM was in the BLOCKED state. This bit is not sticky. It is updated each time a Flash Controller operation is denied. 0 : Flash operation was not stopped due to the BLOCKED state 1 : Flash operation was stopped due to the BLOCKED state Reset type: SYSRSn

### 5.9.4.10 RAMOPENSTAT Register (Offset = 20h) [Reset = 00000000h]

RAMOPENSTAT is shown in [Figure 5-61](#) and described in [Table 5-67](#).

Return to the [Summary Table](#).

32.0

**Figure 5-61. RAMOPENSTAT 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							RAMOPEN
R-0-0h							R-0h

**Table 5-67. RAMOPENSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	RAMOPEN	R	0h	Status of RAM Security after device initialization phase 0 : Normal RAM security rule applies 1: All secured RAMs have become unsecured This bit can be set by the user by writing to the RAMOPENFRC.SET bit. Notes: 1. When this bit is set, previous content gets wiped out by hardware. The RAM data gets initialized with 0x0 data with matching parity/ ECC. 2. This bit can be set only after RAMINIT is over for all the secured RAMs. This bit can be polled by ther user to check completeness of the 'RAM Open' request. 3. While RAMINIT is going on, any access to secured RAMs will get stalled 4. After this bit is set, C28x execution from flash is not allowed Reset type: PORESETn

### 5.9.4.11 RAMOPENFRC Register (Offset = 22h) [Reset = 0000000h]

RAMOPENFRC is shown in [Figure 5-62](#) and described in [Table 5-68](#).

Return to the [Summary Table](#).

32.0

**Figure 5-62. RAMOPENFRC 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															SET
R-0-0h															R-0/ W1S-0 h

**Table 5-68. RAMOPENFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	A write to this register will succeed only when KEY = 0x5a5a. Otherwise the write action is ignored Reset type: N/A
15-1	RESERVED	R-0	0h	Reserved
0	SET	R-0/W1S	0h	User needs to write a '1' to this bit to request the 'RAM Open' mode. When this bit is written with '1', hardware wipes out all the secured RAM content using RAMINIT feature and sets the RAMOPENSTAT.RAMOPEN bit after completion of RAMINIT for all the secured RAMs. Write of '0' is ignored Reset type: N/A

### 5.9.4.12 RAMOPENCLR Register (Offset = 24h) [Reset = 0000000h]

RAMOPENCLR is shown in [Figure 5-63](#) and described in [Table 5-69](#).

Return to the [Summary Table](#).

32.0

**Figure 5-63. RAMOPENCLR 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							CLEAR
R-0-0h							R-0/W1S-0h

**Table 5-69. RAMOPENCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	A write to this register will succeed only when KEY = 0x5a5a. Otherwise the write action is ignored Reset type: N/A
15-1	RESERVED	R-0	0h	Reserved
0	CLEAR	R-0/W1S	0h	User needs to write a '1' to this bit to request the device to come out of 'RAM Open' mode. When this bit is written with '1', hardware wipes out all the secured RAM content using RAMINIT feature and then clears the RAMOPENSTAT.RAMOPEN bit after completion of RAMINIT for all the secured RAMs. Write of '0' is ignored Reset type: N/A

### 5.9.4.13 RAMOPENLOCK Register (Offset = 26h) [Reset = 0000000h]

RAMOPENLOCK is shown in [Figure 5-64](#) and described in [Table 5-70](#).

Return to the [Summary Table](#).

32.0

**Figure 5-64. RAMOPENLOCK Register**

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

**Table 5-70. RAMOPENLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/W1S	0h	If this bit is set, writes to RAMOPENFRC register is not allowed. Once this bit is set, this can only be cleared on an XRSn reset Reset type: XRSn

### 5.9.5 DCSM\_Z1\_OTP Registers

[Table 5-71](#) lists the memory-mapped registers for the DCSM\_Z1\_OTP registers. All register offset addresses not listed in [Table 5-71](#) should be considered as reserved locations and the register contents should not be modified.

**Table 5-71. DCSM\_Z1\_OTP Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	Z1OTP_LINKPOINTER1	Zone 1 Link Pointer1		<a href="#">Go</a>
2h	Z1OTP_LINKPOINTER2	Zone 1 Link Pointer2		<a href="#">Go</a>
4h	Z1OTP_LINKPOINTER3	Zone 1 Link Pointer3		<a href="#">Go</a>
6h	Z1OTP_JLM_ENABLE	Zone 1 JTAGLOCK Enable Register		<a href="#">Go</a>
8h	Z1OTP_GPREG1	Zone 1 General Purpose Register 1		<a href="#">Go</a>
Ah	Z1OTP_GPREG2	Zone 1 General Purpose Register 2		<a href="#">Go</a>
Ch	Z1OTP_GPREG3	Zone 1 General Purpose Register 3		<a href="#">Go</a>
Eh	Z1OTP_GPREG4	Zone 1 General Purpose Register 4		<a href="#">Go</a>
10h	Z1OTP_PSWDLOCK	Secure Password Lock		<a href="#">Go</a>
14h	Z1OTP_JTAGPSWDH0	JTAG Lock Permanent Password 0		<a href="#">Go</a>
16h	Z1OTP_JTAGPSWDH1	JTAG Lock Permanent Password 1		<a href="#">Go</a>
18h	Z1OTP_CMACKKEY0	Secure Boot CMAC Key 0		<a href="#">Go</a>
1Ah	Z1OTP_CMACKKEY1	Secure Boot CMAC Key 1		<a href="#">Go</a>
1Ch	Z1OTP_CMACKKEY2	Secure Boot CMAC Key 2		<a href="#">Go</a>
1Eh	Z1OTP_CMACKKEY3	Secure Boot CMAC Key 3		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 5-72](#) shows the codes that are used for access types in this section.

**Table 5-72. DCSM\_Z1\_OTP Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
Reset or Default Value		
-n		Value after reset or the default value

### 5.9.5.1 Z1OTP\_LINKPOINTER1 Register (Offset = 0h) [Reset = FFFFFFFFh]

Z1OTP\_LINKPOINTER1 is shown in [Figure 5-65](#) and described in [Table 5-73](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer1

**Figure 5-65. 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																															

**Table 5-73. Z1OTP\_LINKPOINTER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER1	R	FFFFFFFh	Zone1 Link Pointer 1 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

### 5.9.5.2 Z1OTP\_LINKPOINTER2 Register (Offset = 2h) [Reset = FFFFFFFFh]

Z1OTP\_LINKPOINTER2 is shown in [Figure 5-66](#) and described in [Table 5-74](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer2

**Figure 5-66. 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																															

**Table 5-74. Z1OTP\_LINKPOINTER2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER2	R	FFFFFFFh	Zone1 Link Pointer 2 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

### 5.9.5.3 Z1OTP\_LINKPOINTER3 Register (Offset = 4h) [Reset = FFFFFFFFh]

Z1OTP\_LINKPOINTER3 is shown in [Figure 5-67](#) and described in [Table 5-75](#).

Return to the [Summary Table](#).

Zone 1 Link Pointer3

**Figure 5-67. 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																															

**Table 5-75. Z1OTP\_LINKPOINTER3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_LINKPOINTER3	R	FFFFFFFh	Zone1 Link Pointer 3 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

### 5.9.5.4 Z1OTP\_JLM\_ENABLE Register (Offset = 6h) [Reset = FFFFFFFFh]

Z1OTP\_JLM\_ENABLE is shown in [Figure 5-68](#) and described in [Table 5-76](#).

Return to the [Summary Table](#).

Zone 1 JTAGLOCK Enable Register

**Figure 5-68. Z1OTP\_JLM\_ENABLE Register**

31	30	29	28	27	26	25	24	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_JLM_ENABLE																															
R-FFFFFFFh																															

**Table 5-76. Z1OTP\_JLM\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_JLM_ENABLE	R	FFFFFFFh	Zone1 JLM_ENABLE register location in USER OTP. Note: When this value is loaded into Z1_JLM_ENABLE, if the value is 32-bit all-1s, the JTAGLOCK will be enabled. Before shipping parts to customers, TI will program the default value to 0xFFFF_000F, which will disable the JTAGLOCK feature. Users should program 0xFFFF_0000 to enable the JTAGLOCK feature. Reset type: N/A

### 5.9.5.5 Z1OTP\_GPREG1 Register (Offset = 8h) [Reset = FFFFFFFFh]

Z1OTP\_GPREG1 is shown in [Figure 5-69](#) and described in [Table 5-77](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register 1

**Figure 5-69. Z1OTP\_GPREG1 Register**

31	30	29	28	27	26	25	24	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_GPREG1																															
R-FFFFFFFh																															

**Table 5-77. Z1OTP\_GPREG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_GPREG1	R	FFFFFFFh	Zone1 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.5.6 Z1OTP\_GPREG2 Register (Offset = Ah) [Reset = FFFFFFFFh]

Z1OTP\_GPREG2 is shown in [Figure 5-70](#) and described in [Table 5-78](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register 2

**Figure 5-70. Z1OTP\_GPREG2 Register**

31	30	29	28	27	26	25	24	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_GPREG2																															
R-FFFFFFFh																															

**Table 5-78. Z1OTP\_GPREG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_GPREG2	R	FFFFFFFh	Zone1 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.5.7 Z1OTP\_GPREG3 Register (Offset = Ch) [Reset = FFFFFFFFh]

Z1OTP\_GPREG3 is shown in [Figure 5-71](#) and described in [Table 5-79](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register 3

**Figure 5-71. Z1OTP\_GPREG3 Register**

31	30	29	28	27	26	25	24	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_GPREG3																															
R-FFFFFFFh																															

**Table 5-79. Z1OTP\_GPREG3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_GPREG3	R	FFFFFFFh	Zone1 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.5.8 Z1OTP\_GPREG4 Register (Offset = Eh) [Reset = FFFFFFFFh]

Z1OTP\_GPREG4 is shown in [Figure 5-72](#) and described in [Table 5-80](#).

Return to the [Summary Table](#).

Zone 1 General Purpose Register 4

**Figure 5-72. Z1OTP\_GPREG4 Register**

31	30	29	28	27	26	25	24	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_GPREG4																															
R-FFFFFFFh																															

**Table 5-80. Z1OTP\_GPREG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_GPREG4	R	FFFFFFFh	Zone1 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.5.9 Z1OTP\_PSWDLOCK Register (Offset = 10h) [Reset = FFFFFFFFh]

Z1OTP\_PSWDLOCK is shown in [Figure 5-73](#) and described in [Table 5-81](#).

Return to the [Summary Table](#).

Secure Password Lock

**Figure 5-73. 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																															

**Table 5-81. Z1OTP\_PSWDLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z1OTP_PSWDLOCK	R	FFFFFFFh	Zone1 password lock location in USER OTP. Note: When this value is loaded into DCSM, if the value is 32-bit all-1s, CSMPSWD will remain locked. Before shipping parts to customers, TI would change the value of this location in such a way that the ECC field remains all-1s and also LSB 4-bits remain 4'b1111. Reset type: N/A

### 5.9.5.10 Z1OTP\_JTAGPSWDH0 Register (Offset = 14h) [Reset = FFFFFFFFh]

Z1OTP\_JTAGPSWDH0 is shown in [Figure 5-74](#) and described in [Table 5-82](#).

Return to the [Summary Table](#).

JTAG Lock Permanent Password 0

**Figure 5-74. Z1OTP\_JTAGPSWDH0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JTAGPSWDH0																															
R-FFFFFFFh																															

**Table 5-82. Z1OTP\_JTAGPSWDH0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	JTAGPSWDH0	R	FFFFFFFh	JTAG Lock Password High 0 (bits 95:64) location in USER Z1 OTP. This value is dummy loaded into the non-memory-mapped JTAGPSWD register, bits 95:64. TI must program a default value into this location, leaving the ECC bits all 1's. Reset type: N/A

### 5.9.5.11 Z1OTP\_JTAGPSWDH1 Register (Offset = 16h) [Reset = FFFFFFFFh]

Z1OTP\_JTAGPSWDH1 is shown in [Figure 5-75](#) and described in [Table 5-83](#).

Return to the [Summary Table](#).

JTAG Lock Permanent Password 1

**Figure 5-75. Z1OTP\_JTAGPSWDH1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JTAGPSWDH1																															
R-FFFFFFFh																															

**Table 5-83. Z1OTP\_JTAGPSWDH1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	JTAGPSWDH1	R	FFFFFFFh	JTAG Lock Password High 1 (bits 127:96) location in USER Z1 OTP. This value is dummy loaded into the non-memory-mapped JTAGPSWD register, bits 127:96. Reset type: N/A

### 5.9.5.12 Z1OTP\_CMACKKEY0 Register (Offset = 18h) [Reset = FFFFFFFFh]

Z1OTP\_CMACKKEY0 is shown in [Figure 5-76](#) and described in [Table 5-84](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key 0

**Figure 5-76. Z1OTP\_CMACKKEY0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMACKKEY0																															
R-FFFFFFFh																															

**Table 5-84. Z1OTP\_CMACKKEY0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CMACKKEY0	R	FFFFFFFh	Secure Boot CMAC Key 0 (bits 31:0) location in User Z1 OTP. This value is dummy loaded into the CMACKKEY0 register. Reset type: N/A

### 5.9.5.13 Z1OTP\_CMACKKEY1 Register (Offset = 1Ah) [Reset = FFFFFFFFh]

Z1OTP\_CMACKKEY1 is shown in [Figure 5-77](#) and described in [Table 5-85](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key 1

**Figure 5-77. Z1OTP\_CMACKKEY1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMACKKEY1																															
R-FFFFFFFh																															

**Table 5-85. Z1OTP\_CMACKKEY1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CMACKKEY1	R	FFFFFFFh	Secure Boot CMAC Key 1 (bits 63:32) location in User Z1 OTP. This value is dummy loaded into the CMACKKEY1 register. Reset type: N/A

#### 5.9.5.14 Z1OTP\_CMACKKEY2 Register (Offset = 1Ch) [Reset = FFFFFFFFh]

Z1OTP\_CMACKKEY2 is shown in [Figure 5-78](#) and described in [Table 5-86](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key 2

**Figure 5-78. Z1OTP\_CMACKKEY2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMACKKEY2																															
R-FFFFFFFh																															

**Table 5-86. Z1OTP\_CMACKKEY2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CMACKKEY2	R	FFFFFFFh	Secure Boot CMAC Key 2 (bits 95:64) location in User Z1 OTP. This value is dummy loaded into the CMACKKEY2 register. Reset type: N/A

### 5.9.5.15 Z1OTP\_CMACKKEY3 Register (Offset = 1Eh) [Reset = FFFFFFFFh]

Z1OTP\_CMACKKEY3 is shown in [Figure 5-79](#) and described in [Table 5-87](#).

Return to the [Summary Table](#).

Secure Boot CMAC Key 3

**Figure 5-79. Z1OTP\_CMACKKEY3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMACKKEY3																															
R-FFFFFFFh																															

**Table 5-87. Z1OTP\_CMACKKEY3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CMACKKEY3	R	FFFFFFFh	Secure Boot CMAC Key 3 (bits 127:96) location in User Z1 OTP. This value is dummy loaded into the CMACKKEY3 register. Reset type: N/A

### 5.9.6 DCSM\_Z2\_OTP Registers

Table 5-88 lists the memory-mapped registers for the DCSM\_Z2\_OTP registers. All register offset addresses not listed in Table 5-88 should be considered as reserved locations and the register contents should not be modified.

**Table 5-88. DCSM\_Z2\_OTP Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	Z2OTP_LINKPOINTER1	Zone 2 Link Pointer1		<a href="#">Go</a>
2h	Z2OTP_LINKPOINTER2	Zone 2 Link Pointer2		<a href="#">Go</a>
4h	Z2OTP_LINKPOINTER3	Zone 2 Link Pointer3		<a href="#">Go</a>
8h	Z2OTP_GPREG1	Zone 2 General Purpose Register 1		<a href="#">Go</a>
Ah	Z2OTP_GPREG2	Zone 2 General Purpose Register 2		<a href="#">Go</a>
Ch	Z2OTP_GPREG3	Zone 2 General Purpose Register 3		<a href="#">Go</a>
Eh	Z2OTP_GPREG4	Zone 2 General Purpose Register 4		<a href="#">Go</a>
10h	Z2OTP_PSWDLOCK	Secure Password Lock		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 5-89 shows the codes that are used for access types in this section.

**Table 5-89. DCSM\_Z2\_OTP Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
Reset or Default Value		
-n		Value after reset or the default value

### 5.9.6.1 Z2OTP\_LINKPOINTER1 Register (Offset = 0h) [Reset = FFFFFFFFh]

Z2OTP\_LINKPOINTER1 is shown in [Figure 5-80](#) and described in [Table 5-90](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer1

**Figure 5-80. 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																															

**Table 5-90. Z2OTP\_LINKPOINTER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER1	R	FFFFFFFh	Zone2 Link Pointer 1 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

### 5.9.6.2 Z2OTP\_LINKPOINTER2 Register (Offset = 2h) [Reset = FFFFFFFFh]

Z2OTP\_LINKPOINTER2 is shown in [Figure 5-81](#) and described in [Table 5-91](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer2

**Figure 5-81. 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																															

**Table 5-91. Z2OTP\_LINKPOINTER2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER2	R	FFFFFFFh	Zone2 Link Pointer 2 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

### 5.9.6.3 Z2OTP\_LINKPOINTER3 Register (Offset = 4h) [Reset = FFFFFFFFh]

Z2OTP\_LINKPOINTER3 is shown in [Figure 5-82](#) and described in [Table 5-92](#).

Return to the [Summary Table](#).

Zone 2 Link Pointer3

**Figure 5-82. 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																															

**Table 5-92. Z2OTP\_LINKPOINTER3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_LINKPOINTER3	R	FFFFFFFh	Zone2 Link Pointer 3 location in USER OTP. Note: [1] ECC comparison is disabled for this location [2] When this value is loaded into DCSM, if the bits[31:14] !=0, device will remain in BLOCKED state. Before shipping parts to customers, TI would change the value of these bits to 0s. Reset type: N/A

#### 5.9.6.4 Z2OTP\_GPREG1 Register (Offset = 8h) [Reset = FFFFFFFFh]

Z2OTP\_GPREG1 is shown in [Figure 5-83](#) and described in [Table 5-93](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register 1

**Figure 5-83. Z2OTP\_GPREG1 Register**

31	30	29	28	27	26	25	24	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_GPREG1																															
R-FFFFFFFh																															

**Table 5-93. Z2OTP\_GPREG1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_GPREG1	R	FFFFFFFh	Zone2 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.6.5 Z2OTP\_GPREG2 Register (Offset = Ah) [Reset = FFFFFFFFh]

Z2OTP\_GPREG2 is shown in [Figure 5-84](#) and described in [Table 5-94](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register 2

**Figure 5-84. Z2OTP\_GPREG2 Register**

31	30	29	28	27	26	25	24	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_GPREG2																															
R-FFFFFFFh																															

**Table 5-94. Z2OTP\_GPREG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_GPREG2	R	FFFFFFFh	Zone2 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.6.6 Z2OTP\_GPREG3 Register (Offset = Ch) [Reset = FFFFFFFFh]

Z2OTP\_GPREG3 is shown in [Figure 5-85](#) and described in [Table 5-95](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register 3

**Figure 5-85. Z2OTP\_GPREG3 Register**

31	30	29	28	27	26	25	24	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_GPREG3																															
R-FFFFFFFh																															

**Table 5-95. Z2OTP\_GPREG3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_GPREG3	R	FFFFFFFh	Zone2 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.6.7 Z2OTP\_GPREG4 Register (Offset = Eh) [Reset = FFFFFFFFh]

Z2OTP\_GPREG4 is shown in [Figure 5-86](#) and described in [Table 5-96](#).

Return to the [Summary Table](#).

Zone 2 General Purpose Register 4

**Figure 5-86. Z2OTP\_GPREG4 Register**

31	30	29	28	27	26	25	24	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_GPREG4																															
R-FFFFFFFh																															

**Table 5-96. Z2OTP\_GPREG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_GPREG4	R	FFFFFFFh	Zone2 General Purpose register location in USER OTP. Reset type: N/A

### 5.9.6.8 Z2OTP\_PSWDLOCK Register (Offset = 10h) [Reset = FFFFFFFFh]

Z2OTP\_PSWDLOCK is shown in [Figure 5-87](#) and described in [Table 5-97](#).

Return to the [Summary Table](#).

Secure Password Lock

**Figure 5-87. 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																															

**Table 5-97. Z2OTP\_PSWDLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	Z2OTP_PSWDLOCK	R	FFFFFFFh	Zone2 password lock location in USER OTP. Note: When this value is loaded into DCSM, if the value is 32-bit all-1s, CSMPWD will remain locked. Before shipping parts to customers, TI would change the value of this location in such a way that the ECC field remains all-1s and also LSB 4-bits remain 4'b1111. Reset type: N/A

### 5.9.7 DCSM Registers to Driverlib Functions

**Table 5-98. DCSM Registers to Driverlib Functions**

File	Driverlib Function
Z1OTP_LINKPOINTER1	
-	
Z1OTP_LINKPOINTER2	
-	
Z1OTP_LINKPOINTER3	
-	
Z1OTP_JLM_ENABLE	
-	
Z1OTP_GPREG1	
-	
Z1OTP_GPREG2	
-	
Z1OTP_GPREG3	
-	
Z1OTP_GPREG4	
-	
Z1OTP_PSWDLOCK	
-	
Z1OTP_JTAGPSWDH0	
-	
Z1OTP_JTAGPSWDH1	
-	
Z1OTP_CMACKEY0	
-	
Z1OTP_CMACKEY1	
-	

**Table 5-98. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>Z1OTP_CMACKKEY2</b>	
-	
<b>Z1OTP_CMACKKEY3</b>	
-	
<b>Z2OTP_LINKPOINTER1</b>	
-	
<b>Z2OTP_LINKPOINTER2</b>	
-	
<b>Z2OTP_LINKPOINTER3</b>	
-	
<b>Z2OTP_GPREG1</b>	
-	
<b>Z2OTP_GPREG2</b>	
-	
<b>Z2OTP_GPREG3</b>	
-	
<b>Z2OTP_GPREG4</b>	
-	
<b>Z2OTP_PSWDLOCK</b>	
-	
<b>Z1_LINKPOINTER</b>	
dcsm.c	DCSM_unlockZone1CSM
dcsm.c	DCSM_readZone1CSMPwd
dcsm.h	DCSM_getZone1LinkPointerError
<b>Z1_OTPSECLOCK</b>	
dcsm.h	DCSM_getZone1OTPSecureLockStatus
<b>Z1_JLM_ENABLE</b>	
-	
<b>Z1_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone1LinkPointerError
<b>Z1_GPREG1</b>	
-	
<b>Z1_GPREG2</b>	
-	
<b>Z1_GPREG3</b>	
-	
<b>Z1_GPREG4</b>	
-	
<b>Z1_CSMKEY0</b>	
dcsm.c	DCSM_unlockZone1CSM
dcsm.c	DCSM_writeZone1CSM
<b>Z1_CSMKEY1</b>	
dcsm.c	DCSM_unlockZone1CSM
dcsm.c	DCSM_writeZone1CSM
<b>Z1_CSMKEY2</b>	

**Table 5-98. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
dcsm.c	DCSM_unlockZone1CSM
dcsm.c	DCSM_writeZone1CSM
<b>Z1_CSMKEY3</b>	
dcsm.c	DCSM_unlockZone1CSM
dcsm.c	DCSM_writeZone1CSM
<b>Z1_CR</b>	
dcsm.h	DCSM_secureZone1
dcsm.h	DCSM_getZone1CSMSecurityStatus
dcsm.h	DCSM_getZone1ControlStatus
<b>Z1_GRABSECT1R</b>	
-	
<b>Z1_GRABSECT2R</b>	
-	
<b>Z1_GRABSECT3R</b>	
-	
<b>Z1_GRABRAM1R</b>	
-	
<b>Z1_EXEONLYSECT1R</b>	
dcsm.c	DCSM_getZone1FlashEXEStatus
<b>Z1_EXEONLYSECT2R</b>	
dcsm.c	DCSM_getZone1FlashEXEStatus
<b>Z1_EXEONLYRAM1R</b>	
dcsm.c	DCSM_getZone1RAMEXEStatus
<b>Z1_JTAGKEY0</b>	
-	
<b>Z1_JTAGKEY1</b>	
-	
<b>Z1_JTAGKEY2</b>	
-	
<b>Z1_JTAGKEY3</b>	
-	
<b>Z1_CMACKKEY0</b>	
-	
<b>Z1_CMACKKEY1</b>	
-	
<b>Z1_CMACKKEY2</b>	
-	
<b>Z1_CMACKKEY3</b>	
-	
<b>Z2_LINKPOINTER</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.c	DCSM_readZone2CSMPwd
dcsm.h	DCSM_getZone2LinkPointerError
<b>Z2_OTPSECLOCK</b>	
dcsm.h	DCSM_getZone2OTPSecureLockStatus

**Table 5-98. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>Z2_LINKPOINTERERR</b>	
dcsm.h	DCSM_getZone2LinkPointerError
<b>Z2_GPREG1</b>	
-	
<b>Z2_GPREG2</b>	
-	
<b>Z2_GPREG3</b>	
-	
<b>Z2_GPREG4</b>	
-	
<b>Z2_CSMKEY0</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.c	DCSM_writeZone2CSM
<b>Z2_CSMKEY1</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.c	DCSM_writeZone2CSM
<b>Z2_CSMKEY2</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.c	DCSM_writeZone2CSM
<b>Z2_CSMKEY3</b>	
dcsm.c	DCSM_unlockZone2CSM
dcsm.c	DCSM_writeZone2CSM
<b>Z2_CR</b>	
dcsm.h	DCSM_secureZone2
dcsm.h	DCSM_getZone2CSMSecurityStatus
dcsm.h	DCSM_getZone2ControlStatus
<b>Z2_GRABSECT1R</b>	
-	
<b>Z2_GRABSECT2R</b>	
-	
<b>Z2_GRABSECT3R</b>	
-	
<b>Z2_GRABRAM1R</b>	
-	
<b>Z2_EXEONLYSECT1R</b>	
dcsm.c	DCSM_getZone2FlashEXEStatus
<b>Z2_EXEONLYSECT2R</b>	
dcsm.c	DCSM_getZone2FlashEXEStatus
<b>Z2_EXEONLYRAM1R</b>	
dcsm.c	DCSM_getZone2RAMEXEStatus
<b>FLSEM</b>	
dcsm.c	DCSM_claimZoneSemaphore
dcsm.c	DCSM_releaseZoneSemaphore
<b>SECTSTAT1</b>	
dcsm.h	DCSM_getFlashSectorZone

**Table 5-98. DCSM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SECTSTAT2</b>	
dcsm.h	DCSM_getFlashSectorZone
<b>SECTSTAT3</b>	
dcsm.h	DCSM_getFlashSectorZone
<b>RAMSTAT1</b>	
dcsm.h	DCSM_getRAMZone
<b>SECERRSTAT</b>	
dcsm.h	DCSM_getFlashErrorStatus
<b>SECERRCLR</b>	
dcsm.h	DCSM_clearFlashErrorStatus
<b>SECERRFRC</b>	
dcsm.h	DCSM_forceFlashErrorStatus
<b>DENYCODE</b>	
dcsm.h	DCSM_getFlashDenyCodeStatus
<b>RAMOPENSTAT</b>	
dcsm.h	DCSM_getRAMOpenStatus
<b>RAMOPENFRC</b>	
dcsm.h	DCSM_forceRAMOpenStatus
<b>RAMOPENCLR</b>	
dcsm.h	DCSM_clearRAMOpenStatus
<b>RAMOPENLOCK</b>	
dcsm.h	DCSM_setRAMOpenLockStatus

This page intentionally left blank.



This chapter describes the Flash module.

<b>6.1 Introduction to Flash and OTP Memory</b> .....	<b>552</b>
<b>6.2 Flash Bank, OTP, and Pump</b> .....	<b>553</b>
<b>6.3 Flash Wrapper</b> .....	<b>554</b>
<b>6.4 Flash and OTP Memory Performance</b> .....	<b>554</b>
<b>6.5 Flash Read Interface</b> .....	<b>555</b>
<b>6.6 Flash Erase and Program</b> .....	<b>558</b>
<b>6.7 Error Correction Code (ECC) Protection</b> .....	<b>558</b>
<b>6.8 Reserved Locations Within Flash and OTP</b> .....	<b>563</b>
<b>6.9 Migrating an Application from RAM to Flash</b> .....	<b>563</b>
<b>6.10 Procedure to Change the Flash Control Registers</b> .....	<b>564</b>
<b>6.11 Software</b> .....	<b>565</b>
<b>6.12 Flash Registers</b> .....	<b>566</b>

## 6.1 Introduction to Flash and OTP Memory

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

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

### 6.1.1 FLASH Related Collateral

#### Foundational Materials

- [C2000 Academy - FLASH](#)
- [Embedded Flash Memory](#) (Video)

#### Getting Started Materials

- [Serial Flash Programming of C2000 Microcontrollers Application Report](#)
- [\[FAQ\] FAQ for Flash ECC usage in C2000 devices - Includes ECC test mode, Linker ECC options:](#)
- [\[FAQ\] FAQ on Flash API usage for C2000 devices](#)
- [\[FAQ\] Flash - How to modify an application from RAM configuration to Flash configuration?](#)
- [\[FAQ\] How can we improve the Flash tool performance?](#)
- [\[FAQ\] TI C2000 Device Programming Tools and Services](#)

### 6.1.2 Features

Features of Flash memory include:

- One Flash bank (Bank0) (refer to the device data sheet for the size of the Flash bank)
- One Flash Wrapper for the Flash bank
- Configurable Flash programming option along with ECC
- Multiple sectors providing the option of leaving some sectors programmed and only erasing specific sectors
- User-programmable OTP locations (in user-configurable DCSM OTP, also called USER OTP) for configuring security, OTP boot-mode and boot-mode select pins (if the user is unable to use the factory-default boot-mode select pins)
- Enhanced performance using the code prefetch mechanism and data cache.
- Configurable wait states to give the best performance for a given execution speed
- Safety Features:
  - SECDED - single-error correction and double-error detection is supported
  - Address bits are included in ECC
  - Test mode to check the health of ECC logic
- Integrated Flash program/erase state machine in the Flash Wrapper
  - Simple Flash API algorithms
  - Fast erase and program times (refer to the device data sheet for details)
- Dual Code Security Module (DCSM) to prevent unauthorized access to the Flash (refer to the *Dual Code Security Module (DCSM)* chapter for details)

### 6.1.3 Flash Tools

Texas Instruments provides the following tools for Flash:

- Code Composer Studio™ (CCS) IDE - the development environment with integrated Flash plugin. TI recommends performing a debug reset and restart after programming the code into Flash using CCS.
- Flash API Library - a set of software peripheral functions to erase/program Flash
- UniFlash - standalone tool to erase/program/verify the Flash content through JTAG. No CCS is required.
- Users must check and install available updates for CCS On-Chip Flash Plugin and UniFlash tools.

### 6.1.4 Default Flash Configuration

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

- Flash Bank and Pump are powered up on device reset.
- ECC is enabled
- Wait-states are set to the maximum (0xF)
- Code-prefetch mechanism and data cache are disabled

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 device that has no password stored in the device, so that Flash programming or loading of code into CSM-protected SRAM can be performed. On devices with a password, this read has no effect and the device remains locked.

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

---

#### Note

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

## 6.2 Flash Bank, OTP, and Pump

This device includes one Flash bank. In addition, there is one-time programmable Flash memory (OTP) in the Flash bank. Flash and OTP are uniformly mapped in both program and data memory space.

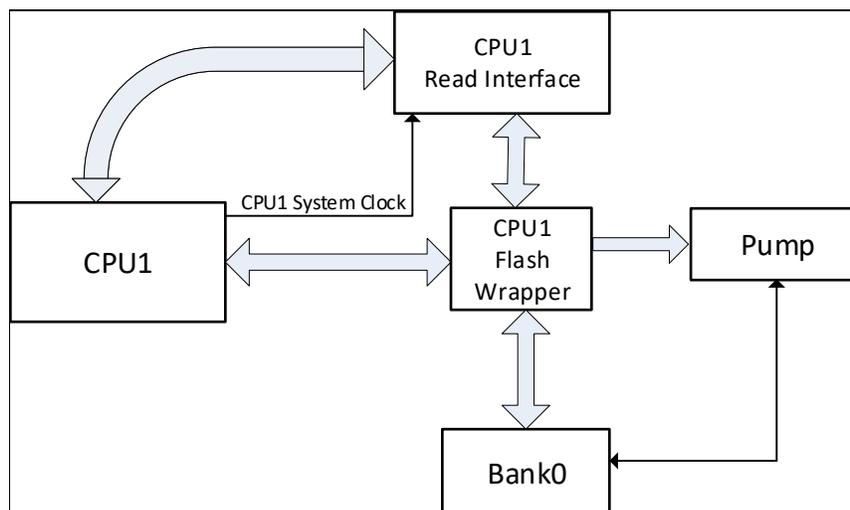
There are two OTP regions. The first OTP region, TI-OTP, contains manufacturing information, trims, Flash operation settings, and other device data. TI-OTP can be read by the user application, but TI-OTP cannot be programmed or erased. The second OTP region, USER-OTP, is primarily used for programming device security (DCSM module) settings. USER-OTP can be programmed only once and cannot be erased afterwards. For information on the memory-map, Flash bank sizes, TI-OTP, USER-OTP, and corresponding ECC locations, refer to the device data sheet.

The *Location of Zone-Select Block Based on Link-Pointer* figure in the *Dual Code Security Module (DCSM)* chapter shows the user-programmable OTP locations in CPU1 USER-OTP. For more information on the functionality of these fields, refer to the *ROM Code and Peripheral Booting* chapter and the *Dual Code Security Module (DCSM)* chapter.

### 6.3 Flash Wrapper

The CPU interfaces with the Flash wrapper, which interfaces with the Flash bank and the pump (see [Figure 6-1](#)). The Flash wrapper has the following primary features:

- Provides a simple interface for software to program or erase the Flash memory.
- Provides an interface for the CPU to read Flash data, including data caching and prefetch features.
- Performs ECC error checking and correction, and generates interrupts when an error is detected.
- Provides the capability to prevent unwanted bank program or erase operations.



**Figure 6-1. Flash Interface Block Diagram**

### 6.4 Flash and OTP Memory Performance

Flash read or instruction fetch accesses can be classified either as a Flash access (access to an address location in Flash), or an OTP access (access to an address location in OTP).

When the CPU performs an access to a Flash memory address, data is returned after (RWAIT+1) 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 configured using the RWAIT field in the FRDCNTL register. At reset, RWAIT defaults to a worst-case wait state count (15), and therefore must be initialized to the appropriate number of wait states to improve performance, based on the CPU clock frequency and the access time of the Flash. The Flash supports zero-wait accesses when RWAIT is set to zero, when the CPU clock frequency is low enough to accommodate the Flash access time.

For a given system clock frequency, configure RWAIT using the following formula:

For C28x Flash Bank:  $RWAIT = \text{ceiling}[(SYSCLK/FCLK)-1]$

where SYSCLK is the system operating frequency for CPU1, and FCLK is the clock frequency for Flash.

FCLK must be  $\leq FCLK_{max}$ , the allowed maximum Flash clock frequency at RWAIT=0.

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

## 6.5 Flash Read Interface

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

### 6.5.1 C28x-Flash Read Interface

#### 6.5.1.1 Standard Read Mode

Standard read mode is the default Flash read mode after reset. In this mode, the code prefetch mechanism and data cache are disabled. When standard read mode is active, every read access to Flash is decoded by the Flash wrapper to fetch the data from the addressed location, and the data is returned after  $RWAIT+1$  cycles (except User OTP).

Flash data buffers associated with the 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, where  $RWAIT$  can be set to zero to provide single-cycle access operation. The Flash wrapper 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 the data cache and prefetch mechanisms to improve performance. Refer to the device data sheet to determine the maximum Flash frequency allowed in standard read mode (that is, maximum Flash clock frequency with  $RWAIT=0$ ,  $FCLK_{MAX}$ ).

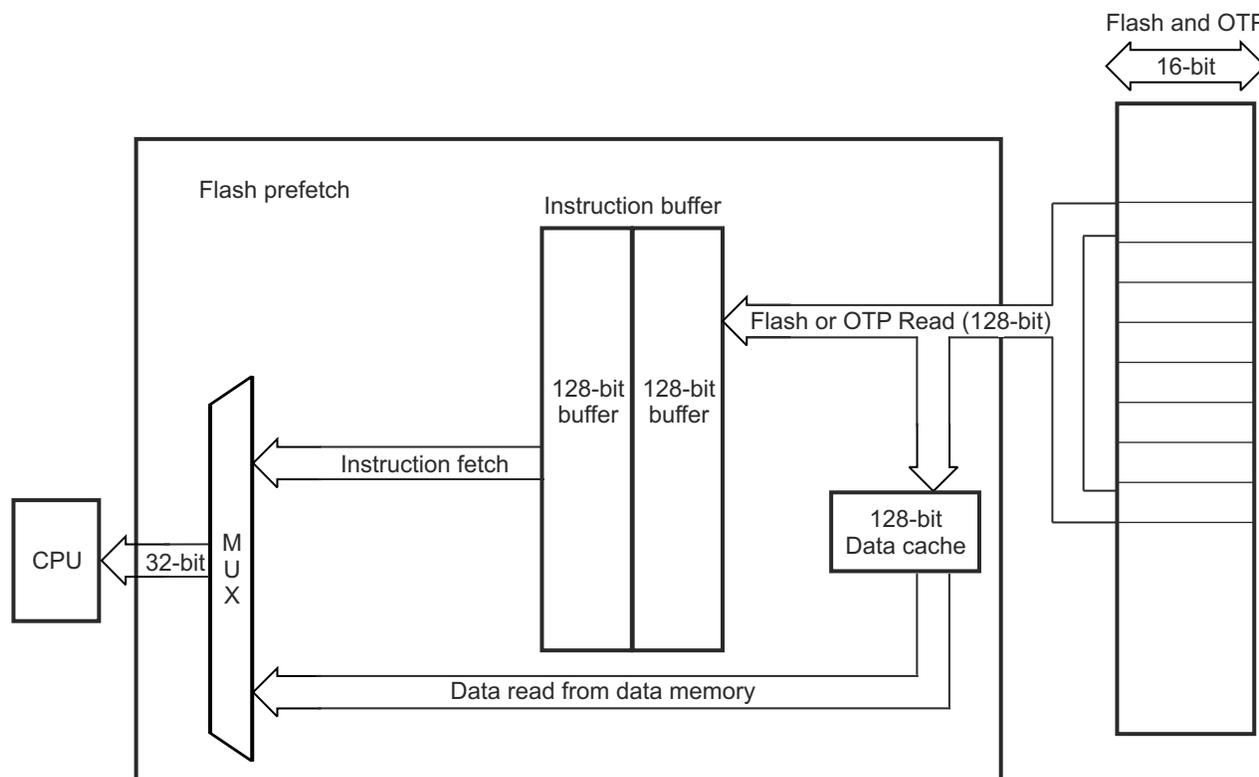
#### 6.5.1.2 Prefetch Mode

Flash memory is typically used to store application code. During code execution, instructions are fetched from contiguous memory addresses, except when a discontinuity occurs. Usually, the portion of the code that resides in contiguous address locations 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. [Figure 6-2](#) illustrates how this mode functions.

The 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. To enable prefetch mode, set the `PREFETCH_EN` bit in the `FRD_INTF_CTRL` register, or call the `Flash_enablePrefetch()` driverlib function.

Each instruction fetch from the Flash or OTP reads out 128 bits. 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. When Flash prefetch mode is 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 there are up to eight instructions in the prefetch buffer ready to process through the CPU. During the time to process these instructions, the Flash prefetch mechanism automatically initiates another access to the Flash bank to prefetch the next 128 bits. 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.



**Figure 6-2. Flash Prefetch Mode**

The Flash prefetch is aborted only when there is a code discontinuity caused by executing an instruction such as a branch, function 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 the code branches 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 mechanism and thus bypass the prefetch buffer. For example, instructions such as MAC, DMAC, and PREAD read a data value from program memory. When such a 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 is stalled until the prefetch completes.

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

### 6.5.1.3 Data Cache

In addition to the prefetch mechanism, a data cache of 128-bits wide has been implemented to improve data space read performance. This data cache is separate from the instruction prefetch buffer, and is used for data reads only. Whenever a data read access is performed by the CPU to a Flash bank address, if the data located at that address is not presently loaded into the data cache, then the Flash wrapper reads 128 bits of data from the Flash bank and stores the data in the data cache. This data is eventually sent to the CPU for processing. The starting address of the Flash bank access 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.

The data cache is disabled by default at reset. To enable the data cache, set the DATA\_CACHE\_EN bit in the FRD\_INTF\_CTRL register, or call the Flash\_enableCache() driverlib function. Note that the data cache gets bypassed when RWAIT is set to zero.

---

#### Note

The data cache does not get updated on a debugger access, or when a read to the ECC memory-mapped region is performed.

---

### 6.5.1.4 Flash Read Operation

There are a few important points to keep in mind when using Flash or OTP memory:

- Reads of USER OTP locations are hardwired for 9 wait states. The RWAIT bits have no effect on these locations.
- CPU writes to Flash or OTP memory addresses are ignored, and complete within a single cycle. Flash memory can only be modified by issuing program or erase commands, using the Flash API.
- When a security zone is in the locked state, and the respective password lock bits are not all ones, then:
  - Data reads to Zx-CSMPSWD return zero;
  - Program space reads to Zx-CSMPSWD return zero; and
  - Program fetches to Zx-CSMPSWD return zero.
- When the Code Security Module (CSM) is secured, reads to Flash or OTP memory addresses from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns zero.
- The arbitration scheme in the Flash wrapper prioritizes CPU accesses in the fixed priority order of data space read (highest priority), program space read, and program fetches/program prefetches (lowest priority).
- When Flash state machine is activated for erase or program operations, the contents of the prefetch buffer and data cache are automatically flushed.

---

#### Note

ECC checks are performed on data read from Flash before the data is stored in the prefetch buffer or data cache. Once data has entered the cache or buffer, there are no further ECC checks performed.

---

## 6.6 Flash Erase and Program

Flash memory can be programmed either by using the CCS Flash plug-in or by using the UniFlash application. If these methods are not feasible in an application, the Flash API can be used. The Flash memory can be programmed, erased, and verified only by using the 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.

The recommended flow for programming Flash is:

Erase → Program → Verify

### 6.6.1 Erase

When the target Flash is erased, the Flash reads as all 1s. This state is called 'blank.' The erase function must be executed before programming. The user cannot skip erase on sectors that read as 'blank' because these sectors can 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. Bank erase is also supported in this device.

### 6.6.2 Program

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

---

#### Note

The main array Flash programming must be aligned to 64-bit address boundaries, and each 64-bit word can only be programmed once per write/erase cycle.

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

- The DCSM Zx-LINKPOINTER1 and Zx-LINKPOINTER2 values in the DCSM OTP can be programmed together and can be programmed one bit at a time as required by the DCSM operation.
- The DCSM Zx-LINKPOINTER3 values in the DCSM OTP can be programmed one bit at a time as required by the DCSM operation.

To avoid exceeding data retention capability limits, do not perform more than 64 program operations on the same Flash word line before performing an erase operation. Each Flash word line consists of sixteen 128-bit words (256 bytes). This limit is especially important to observe when writing to one-time-programmable Flash regions, such as the User OTP.

---

### 6.6.3 Verify

After programming, the user must perform verify using API function `Fapi_doVerify()`. This function verifies the Flash contents against supplied data.

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

## 6.7 Error Correction Code (ECC) Protection

There are two ECC blocks (ECC64\_H and ECC64\_L) inside the Flash Read Interface. These ECC blocks correct single-bit Flash read errors, and can detect uncorrectable errors of two or more bits. The ECC blocks are also capable of detecting address errors. The ECC blocks operate using eight user-calculated ECC check bits associated with each 64-bit wide data word and the corresponding 128-bit memory-aligned address. Users must

program these ECC check bits into ECC memory space, along with Flash data during the Flash programming operation. Refer to the device data sheet for the Flash/OTP ECC memory-map.

The ECC bits for a given Flash address and 64-bit data word can be calculated using the Flash API; however, TI recommends using the AutoEccGeneration option available in the Flash Plugin or API to auto-calculated and program ECC bits. The 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 the Code Composer Studio™ IDE, uses the Flash API to generate and program ECC data.

Figure 6-3 illustrates the ECC logic inputs and outputs.

During an instruction fetch or a data read operation, the 19 most-significant address bits (the 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 ECC logic, and the eight check bits are produced in ECC block. 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 ECC block 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

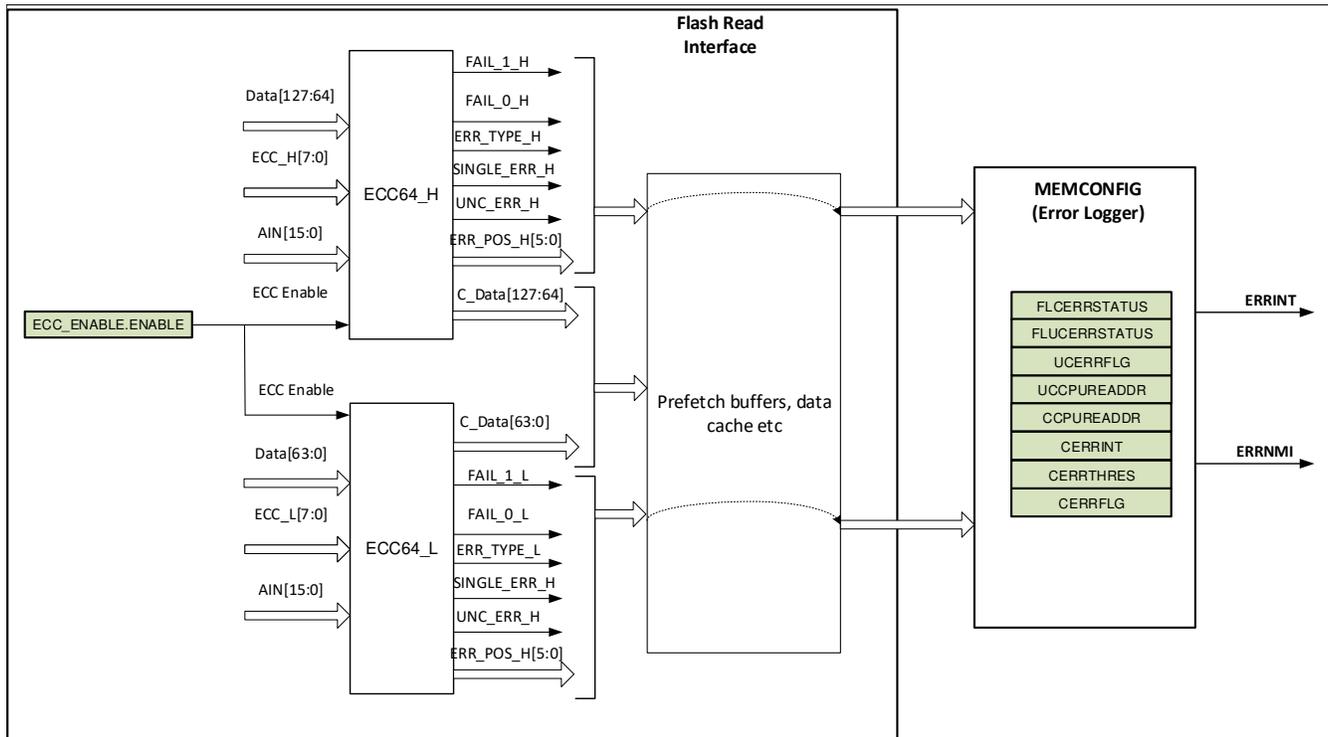


Figure 6-3. ECC Logic Inputs and Outputs

A single-bit error in the address field is considered to be a non-correctable error.

**Note**

Since ECC is calculated for an entire 64-bit data word, a non 64-bit read such as a byte read or a half-word read still forces the entire 64-bit data word to be read and calculated, even though only the byte or half-word is actually used by the CPU.

The ECC feature is enabled by default at reset, and can be enabled or disabled by writing to the ECC\_ENABLE register. ECC logic is automatically bypassed when the 64 data bits and associated ECC bits fetched from the bank are either all ones or all zeros.

### 6.7.1 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 the error 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 Flash data word, the address of the lower 64-bit word is captured in the SINGLE\_ERR\_ADDR\_LOW register. If the single-bit error occurs in the upper 64 bits of the 128-bit data word, then the address of the upper 64-bit word is 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 Flash data word.
- Bit position at which the 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 Flash data word.
- 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 ERR\_THRESHOLD+1, and a single bit error occurs, the Flash module sets the SINGLE\_ERR\_INT flag and generates an interrupt signal. To enable propagation of the generated interrupt pulse to the CPU, the user application must enable the FLASH\_CORRECTABLE\_ERROR channel in the C28 Peripheral Interrupt Expansion module (PIE). The interrupt signal remains high until the application clears the SINGLE\_ERR\_INTFLG flag by writing to the SINGLE\_ERR\_INTCLR bit in the ERR\_INTCLR register. The Flash module cannot generate any further FLASH\_CORRECTABLE\_ERROR interrupt signals to the PIE/CPU until SINGLE\_ERR\_INTFLG is cleared, as this is an edge-based interrupt.

When multiple single-bit errors have been detected by ECC logic, the contents of the Flash ECC registers reflect the most recent ECC error. When multiple single-bit errors have been detected, both FAIL\_0\_L and FAIL\_1\_L (or FAIL\_0\_H and FAIL\_1\_H) can be set, indicating that single-bit fail0/fail1 occurred in different 64-bit aligned addresses.

Although ECC is calculated on 64-bit basis, a read of any address location within a 128-bit aligned Flash data word causes the single-bit error flag to get set, if there is a single-bit error in both or in either the lower 64 or upper 64 bits (or corresponding ECC check bits) of that 128-bit data word.

### 6.7.2 Uncorrectable Error

Uncorrectable errors include address errors and double-bit errors in data or ECC. When the ECC logic 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 Flash data word, the lower 64-bit memory-aligned address is captured in the `UNC_ERR_ADDR_LOW` register. If the uncorrectable error occurs in the upper 64 bits of a 128-bit memory-aligned Flash data word, the upper 64-bit memory-aligned address is 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 Flash data word.
- A flag is set indicating that an uncorrectable error interrupt is generated (`UNC_ERR_INTFLG` in `ERR_INTFLG` register).

When an uncorrectable error occurs, the Flash module sets the `UNC_ERR_INTFLG` bit and generates an uncorrectable error interrupt. This uncorrectable error interrupt generates a non-maskable interrupt (NMI), if enabled, in the CPU. If an uncorrectable error interrupt flag is not cleared by writing to the `UNC_ERR_INTCLR` bit in the `ERR_INTCLR` register, the Flash module cannot generate new uncorrectable interrupt signals, 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 word causes the uncorrectable error flag to get set, and an uncorrectable error interrupt/NMI to occur, when there is a uncorrectable error in both or in either the lower 64 bits or upper 64 bits (or corresponding ECC check bits) of that 128-bit data word.

### 6.7.3 Mechanism to Check the Correctness of ECC Logic

To make sure the correctness of the ECC logic, a redundant ECC logic block for each of the `ECC64_L` and `ECC64_H` checkers is used. Each 64-bit ECC checker block and the corresponding redundant checker block receive the same inputs. The output of each 64-bit checker block is bitwise XORed with the output of the corresponding redundant checker block; a non-zero output from this comparison generates an uncorrectable error (`UNC_ERR`) signal. This redundancy makes sure that any fault in ECC logic circuits can be detected and trigger an NMI.

A mechanism has been added to enable self-testing of the ECC logic for additional diagnostic coverage. To use this mechanism, configure the `ECC_TEST_EN` field in the `FECC_CTRL` register. A value of 01 in `ECC_TEST_EN` injects a single-bit error into the redundant ECC logic upon a Flash read access, and a value of 10 injects a double-bit error upon a Flash read access. This causes an output comparison failure. In this mode, the diagnostic outputs of each of the high and low comparators (`DIAG_H` and `DIAG_L`) are captured in the `FLUCERRSTATUS` Memconfig register.

---

#### Note

When ECC self-test is enabled and CPU issues a read access to the Flash, ECC errors are captured in the data cache and prefetch buffers. TI recommends that application software disables caching while performing diagnostic checks.

---

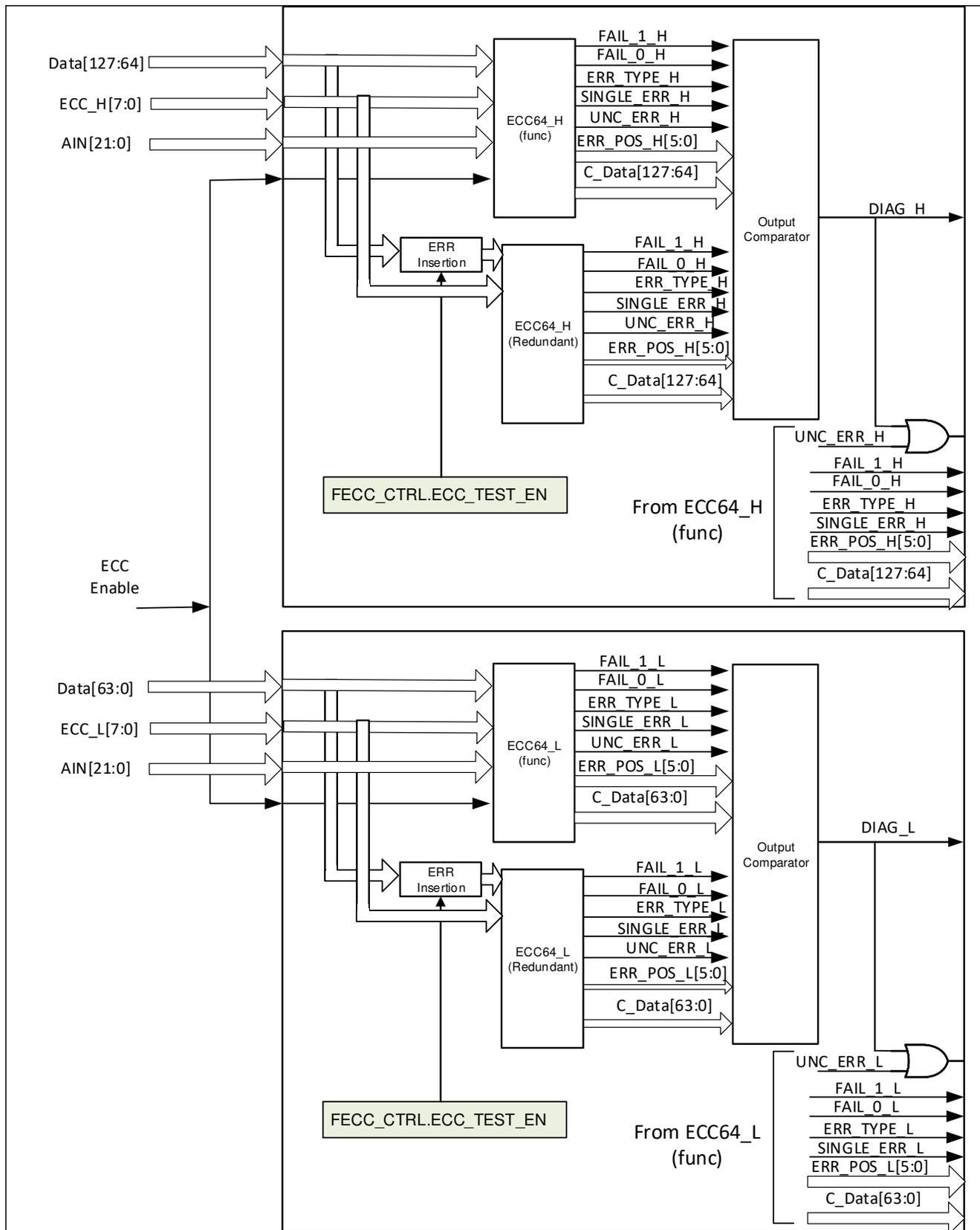


Figure 6-4. Testing ECC Logic

## 6.8 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, refer to the *ROM Code and Peripheral Booting* chapter and the *Dual Code Security Module (DCSM)* 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 jumps to this address in Flash. If you program a branch instruction here, that redirects code execution to the entry point of the application.

## 6.9 Migrating an Application from RAM to Flash

To migrate an existing application that is configured to run from RAM to a Flash-based linker configuration, follow these steps:

1. Replace the RAM linker command file with a Flash linker command file. For examples of Flash-based linker command files, see the `device_support\ directory.`
2. When modifying the Flash-based linker command file, be sure to map any initialized sections to Flash memory regions.
3. Make sure the boot mode pins are configured for Flash boot. This tells the boot ROM to redirect execution to the application programmed into Flash memory after boot code execution is complete. For more information on boot mode configuration, see *Detailed Description > Device Boot Modes* in the device data sheet.
4. When the device is configured for Flash boot, the boot ROM redirects execution to the Flash entry point location (defined as BEGIN in TI-provided Flash linker command files) at the end of boot code execution. Make sure there is a branch instruction at the Flash entry point to your code initialization (for example, `_c_int00`) function. In the C2000Ware examples, the entry point code is specified in the `codestartbranch.asm` file.
5. To achieve best performance for Flash execution, configure the Flash wait states as per the device operating clock frequency, as specified in the device data sheet. In addition, enable prefetch mode and data cache mode. Calling the `Flash_initModule()` driverlib function achieves these steps. Note that code that initializes the Flash module must execute from a RAM location. This is accomplished by assigning the Flash initialization function to the `.TI.ramfunc` section. In the linker command file, map this section to Flash for load, and RAM for execution. The example cmd files provided in C2000Ware show how to do this correctly.
6. For any functions that require 0- or 1-wait state performance, be sure to map to RAM for execution in the linker command file, similar to the Flash initialization function. The `.TI.ramfunc` section in the TI-provided Flash linker command files accomplishes this purpose.
7. Align all code and data sections to 128-bit address boundaries when mapping to Flash memory, using the ALIGN directive in the linker command file.
8. For EABI executable formats, define all uninitialized sections mapped to RAM as NOINIT sections (using the directive "`type=NOINIT`") in the linker command file.
9. Be sure to program ECC bits correctly for the Flash application image. Keep the AutoEccGeneration option enabled in the Code Composer Studio Flash Plugin or UniFlash GUI.

## 6.10 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, 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 and must reside in RAM.
3. Execute the Flash configuration code (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 that resides in RAM or Flash/OTP and continue execution.

## 6.11 Software

### 6.11.1 FLASH Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/flash

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 6.11.1.1 Flash Programming with AutoECC, DataAndECC, DataOnly and EccOnly

FILE: flashapi\_ex1\_programming.c

This example demonstrates how to program Flash using API's following options

1. AutoEcc generation
2. DataOnly and EccOnly
3. DataAndECC

##### *External Connections*

- None.

##### *Watch Variables*

- None.

#### 6.11.1.2 Boot Source Code

FILE: flash\_kernel\_ex3\_boot.c

Functions: void copyData(void) uint32\_t getLongData(void) void readReservedFn(void)

#### 6.11.1.3 Erase Source Code

FILE: flash\_kernel\_ex3\_erase.c Functions:

#### 6.11.1.4 Live DFU Command Functionality

FILE: flash\_kernel\_ex3\_ldfu.c

#### 6.11.1.5 Verify Source Code

FILE: flash\_kernel\_ex3\_verify.c

#### 6.11.1.6 SCI Boot Mode Routines

FILE: flash\_kernel\_ex3\_sci\_boot.c Functions: uint32\_t sciBoot(void) void scialnit(void) uint32\_t sciaGetWordData(void)

#### 6.11.1.7 Flash Programming Solution using SCI

FILE: flash\_kernel\_ex3\_sci\_flash\_kernel.c

In this example, we set up a UART connection with a host using SCI, receive commands for CPU1 to perform which then sends ACK, NAK, and status packets back to the host after receiving and completing the tasks. This kernel has the ability to program, verify, unlock, reset, and run an application. Each command either expects no data from the command packet or specific data relative to the command.

In this example, we set up a UART connection with a host using SCI, receive an application for CPU01 in -sci8 ascii format to run on the device and program it into Flash.

## 6.12 Flash Registers

This section describes the Flash Module Registers.

### 6.12.1 FLASH Base Address Table

**Table 6-1. FLASH Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
Flash0CtrlRegs	FLASH_CTRL_REGS	FLASH0CTRL_BASE	0x0005_F800	YES
Flash0EccRegs	FLASH_ECC_REGS	FLASH0ECC_BASE	0x0005_FB00	YES

## 6.12.2 FLASH\_CTRL\_REGS Registers

Table 6-2 lists the memory-mapped registers for the FLASH\_CTRL\_REGS registers. All register offset addresses not listed in Table 6-2 should be considered as reserved locations and the register contents should not be modified.

**Table 6-2. FLASH\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	FRDCNTL	Flash Read Control Register	EALLOW	<a href="#">Go</a>
4h	FLPROT	Flash program/erase protect register	EALLOW	<a href="#">Go</a>
180h	FRD_INTF_CTRL	Flash Read Interface Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 6-3 shows the codes that are used for access types in this section.

**Table 6-3. FLASH\_CTRL\_REGS Access Type Codes**

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

### 6.12.2.1 FRDCNTL Register (Offset = 0h) [Reset = 0F000F00h]

FRDCNTL is shown in [Figure 6-5](#) and described in [Table 6-4](#).

Return to the [Summary Table](#).

Flash Read Control Register

**Figure 6-5. FRDCNTL Register**

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

**Table 6-4. FRDCNTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved
27-24	RESERVED	R/W	Fh	Reserved
23-12	RESERVED	R	0h	Reserved
11-8	RWAIT	R/W	Fh	Random read waitstate These bits indicate how many waitstates are added to a flash read/ fetch access. The RWAIT value can be set anywhere from 0 to 0xF. For a flash access, data is returned in RWAIT+1 SYSCLK cycles. Note: The required wait states for each SYSCLK frequency can be found in the device data manual. Reset type: SYSRSn
7-0	RESERVED	R	0h	Reserved

### 6.12.2.2 FLPROT Register (Offset = 4h) [Reset = 0000000h]

FLPROT is shown in [Figure 6-6](#) and described in [Table 6-5](#).

Return to the [Summary Table](#).

Flash program/erase protect register

**Figure 6-6. FLPROT 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							FLWEPROT
R-0h							R/W-0h

**Table 6-5. FLPROT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	FLWEPROT	R/W	0h	Flash program/erase protect bit. 0 : Program erase operation allowed subject to security settings. 1 : Program erase operation blocked in hardware. Reset type: SYSRSn

### 6.12.2.3 FRD\_INTF\_CTRL Register (Offset = 180h) [Reset = 0000000h]

FRD\_INTF\_CTRL is shown in [Figure 6-7](#) and described in [Table 6-6](#).

Return to the [Summary Table](#).

Flash Read Interface Control Register

**Figure 6-7. FRD\_INTF\_CTRL Register**

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

**Table 6-6. FRD\_INTF\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1	DATA_CACHE_EN	R/W	0h	Data cache enable. 0 A value of 0 disables the data cache. 1 A value of 1 enables the data cache. Reset type: SYSRSn
0	PREFETCH_EN	R/W	0h	Prefetch enable. 0 A value of 0 disables prefetch mechanism. 1 A value of 1 enables pre-fetch mechanism. Reset type: SYSRSn

### 6.12.3 FLASH\_ECC\_REGS Registers

Table 6-7 lists the memory-mapped registers for the FLASH\_ECC\_REGS registers. All register offset addresses not listed in Table 6-7 should be considered as reserved locations and the register contents should not be modified.

**Table 6-7. FLASH\_ECC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ECC_ENABLE	ECC Enable	EALLOW	<a href="#">Go</a>
20h	FECC_CTRL	ECC Control	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 6-8 shows the codes that are used for access types in this section.

**Table 6-8. FLASH\_ECC\_REGS Access Type Codes**

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

### 6.12.3.1 ECC\_ENABLE Register (Offset = 0h) [Reset = 000000Ah]

ECC\_ENABLE is shown in [Figure 6-8](#) and described in [Table 6-9](#).

Return to the [Summary Table](#).

ECC Enable

**Figure 6-8. ECC\_ENABLE Register**

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

**Table 6-9. ECC\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-4	RESERVED	R	0h	Reserved
3-0	ENABLE	R/W	Ah	ECC enable. A value of 0xA would enable ECC. Any other value would disable ECC. Reset type: SYSRSn

### 6.12.3.2 FECC\_CTRL Register (Offset = 20h) [Reset = 0000000h]

FECC\_CTRL is shown in Figure 6-9 and described in Table 6-10.

Return to the [Summary Table](#).

ECC Control

**Figure 6-9. 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						ECC_TEST_EN	
R-0h						R/W-0h	

**Table 6-10. FECC\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-2	RESERVED	R	0h	Reserved
1-0	ECC_TEST_EN	R/W	0h	ECC test mode enable. 00 ECC test mode disabled 01 ECC test mode enabled, one of the 64 data bits is flipped and fed to the redundant ECC logic (on both ECC logic low and ECC logic high blocks). 11 ECC test mode enabled, Two of the 64 data bits are flipped and fed to the redundant ECC logic (on both ECC logic low and ECC logic high blocks). 10 Reserved Reset type: SYSRSn

### 6.12.4 FLASH Registers to Driverlib Functions

**Table 6-11. FLASH Registers to Driverlib Functions**

File	Driverlib Function
<b>FRDCNTL</b>	
flash.h	Flash_setWaitstates
<b>FLPROT</b>	
flash.h	Flash_setFLWEPROT
<b>FRD_INTF_CTRL</b>	
flash.h	Flash_enablePrefetch
flash.h	Flash_disablePrefetch
flash.h	Flash_enableCache
flash.h	Flash_disableCache
<b>ECC_ENABLE</b>	
flash.h	Flash_enableECC
flash.h	Flash_disableECC

**Table 6-11. FLASH Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>FECC_CTRL</b>	
flash.h	Flash_enableSingleBitECCTestMode
flash.h	Flash_enableDoubleBitECCTestMode
flash.h	Flash_disableSingleBitECCTestMode
flash.h	Flash_disableDoubleBitECCTestMode

Chapter 7  
**Dual-Clock Comparator (DCC)**

---



This chapter describes the Dual-Clock Comparator (DCC) module.

<b>7.1 Introduction</b> .....	<b>576</b>
<b>7.2 Module Operation</b> .....	<b>577</b>
<b>7.3 Interrupts</b> .....	<b>583</b>
<b>7.4 Software</b> .....	<b>584</b>
<b>7.5 DCC Registers</b> .....	<b>586</b>

## 7.1 Introduction

The dual-clock comparator module is used for evaluating and monitoring the clock input based on a second clock, which can be a more accurate and reliable version. This instrumentation is used to detect faults in clock source or clock structures, thereby enhancing the system's safety metrics.

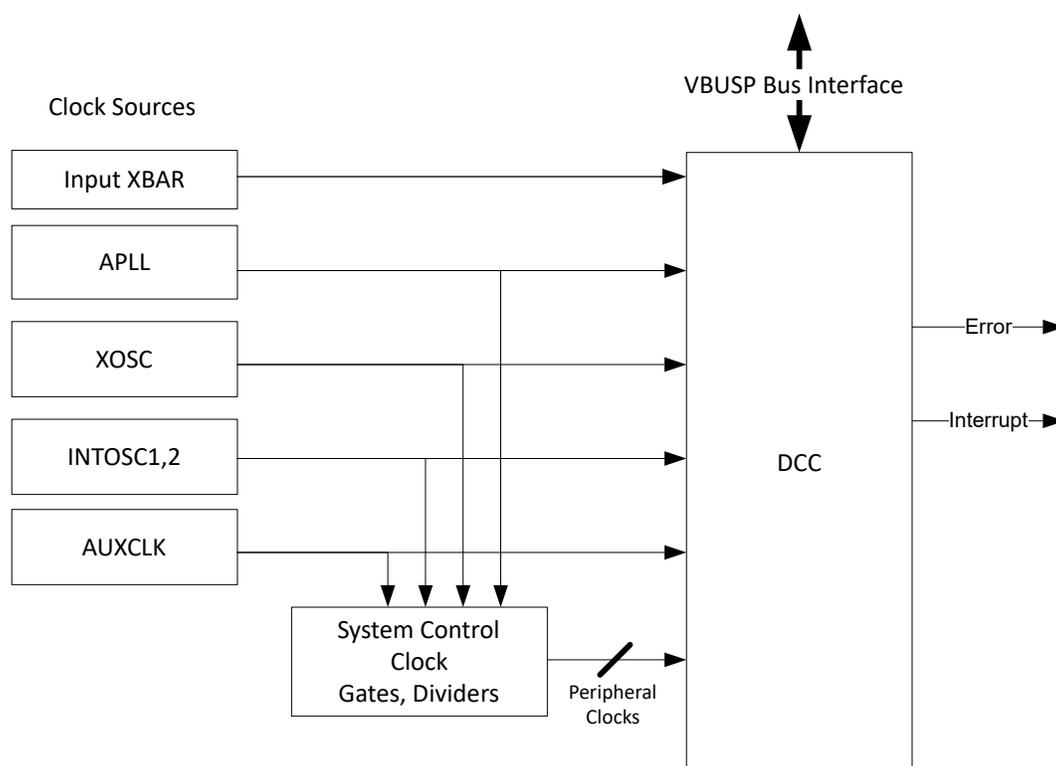
### 7.1.1 Features

The main features of each of the DCC modules are:

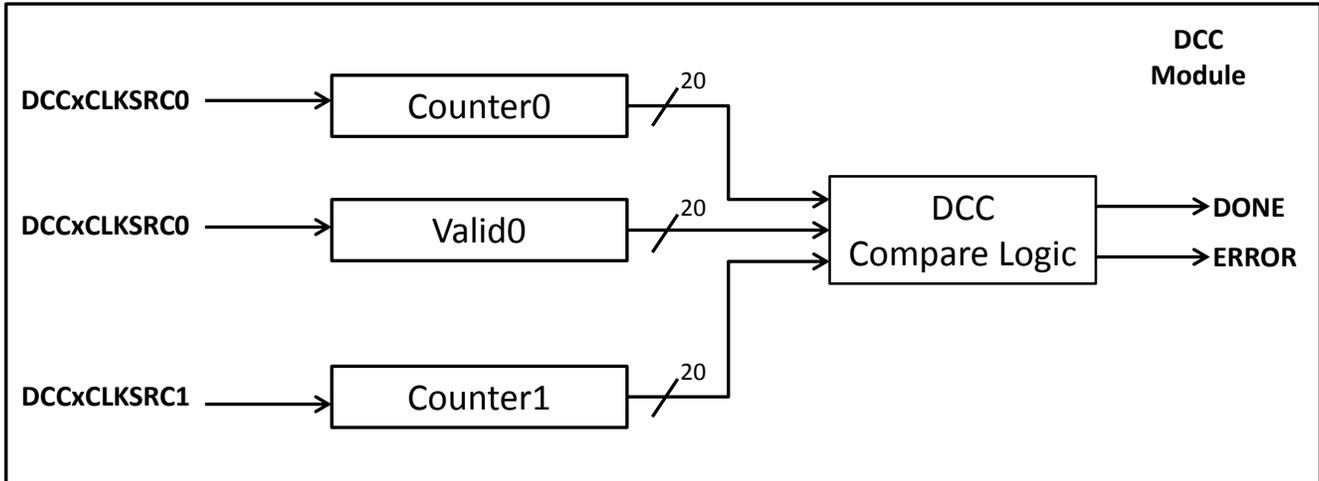
- Allows the application to make sure that a fixed ratio is maintained between frequencies of two clock signals.
- Supports the definition of a programmable tolerance window in terms of the number of reference clock cycles.
- Supports continuous monitoring without requiring application intervention.
- Supports a single-sequence mode for spot measurements.
- Allows the selection of a clock source for each of the counters, resulting in several specific use cases.

### 7.1.2 Block Diagram

Figure 7-1 shows how the DCC connects to the rest of the system. Figure 7-2 shows the main concept of the DCC module.



**Figure 7-1. DCC Module Overview**



**Figure 7-2. DCC Operation**

### 7.2 Module Operation

As shown in [Figure 7-2](#), DCC contains three counters – Counter0, Valid0 and Counter1. Initially, all counters are loaded with the user-defined, pre-load value. Counter0 and Counter1 start decrementing once the DCC is enabled at rates determined by the frequencies of Clock0 and Clock1, respectively. When Counter0 equals 0 (expires), the Valid0 counter decrements at a rate determined by Clock0. If Counter1 decrements to 0 in the valid window, then no error is generated and Clock1 is considered to be good within allowable tolerance as configured by the user.

### 7.2.1 Configuring DCC Counters

Counter0 and Counter1 are configured based on the ratio between the frequencies of Clock0 and Clock1 ( $F_{clk1} \times \text{Counter0} = F_{clk0} \times \text{Counter1}$ ). The Valid0 counter provides tolerance and is configured based on the error in DCC. Since Clock0 and Clock1 are asynchronous, the start and stop of the counters do not occur synchronously. Hence, while configuring the counters, two different sources of errors must be accounted for:

- DCC Errors due to the asynchronous timing of Clock0 and Clock1: this depends on the frequency of Clock0 and Clock1:
  - If  $F_{clk1} > F_{clk0}$ , then Async. Error (in Clock0 cycles) =  $2 + 2 \times (F_{sysclk} / F_{clk0})$
  - If  $F_{clk1} < F_{clk0}$ , then Async. Error (in Clock0 cycles) =  $2 \times (F_{clk0} / F_{clk1}) + 2 \times (F_{sysclk} / F_{clk0})$
  - If  $F_{clk1}$  is unknown, then Async. Error (in Clock0 cycles) =  $2 + 2 \times (F_{sysclk} / F_{clk0})$
- Digitization Error = 8 Clock0 cycles

#### DCC Error (in Clock0 Cycles) = Async. Error + Digitization Error

DCC error shows up as a frequency error for clock under measurement. This error is DCC induced and does not represent error in frequency of clock under measurement. The application needs to take this into consideration while configuring the counters, and determine a desirable tolerance for DCC error that defines the window of measurement. To illustrate:

#### Window (in Clock0 Cycles) = (DCC Error) / (0.01 × Tolerance)

For example, if DCC Error is 10 and the tolerance desired is +/-0.1%, then:

$$\text{Window (in Clock0 Cycles)} = 10 / (0.01 \times 0.1) = 10000$$

Based on above formula for Window, if the desired tolerance is low, then the counter values are large and increase the window of measurement. This means that counter values for a tolerance of 0.1% are larger than that of 0.2%. So, based on the application defined tolerance, define the window of measurement in terms of Clock0 cycles.

The clock under measurement can have an allowed frequency error. If this error is expected, then the error can also be accounted while configuring counters. For example, if measuring INTOSC1/2 frequency using an external crystal as a reference clock, the allowable tolerance of INTOSC1/2 (for example, +/-1%) can be accounted for and factored into the counter configuration. The formula is:

$$\text{Frequency Error Allowed (in Clock0 Cycles)} = \text{Window} \times (\text{Allowable Frequency Tolerance (in \%)} / 100)$$

$$\text{Total Error (in Clock0 Cycles)} = \text{DCC Error} + \text{Frequency Error Allowed}$$

The following equations are used to configure counter values:

$$\text{Counter0 (DCCNTSEED0)} = \text{Window} - \text{Total Error}$$

$$\text{Valid0 (DCCVALIDSEED0)} = 2 \times \text{Total Error}$$

$$\text{Counter1 (DCCNTSEED1)} = \text{Window} \times (F_{clk1} / F_{clk0})$$

---

#### Note

Counter1 is a 20-bit counter, so the maximum possible value cannot exceed 1048575. If the value does exceed, then increase the desired Tolerance for DCC error, so that Window of measurement is lowered. The following formula can be used to compute minimum tolerance possible:

$$\text{Tolerance (\%)} = (100 \times \text{DCC Error} \times (F_{clk1} / F_{clk0})) / 1048575$$


---

## 7.2.2 Single-Shot Measurement Mode

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting. This mode is typically used for spot-checking the frequency of a signal.

### Example-1: Validating PLLRAWCLK frequency

A practical example of the usage is to validate the PLL output clock frequency using the XTAL as the reference clock. Assume XTAL is 10MHz, PLL output frequency is 100MHz, SYSCLK is 100MHz, allowable Frequency Tolerance is 0.1%, and DCC Tolerance required is 0.1%. The measurement sequence proceeds as follows:

- Set Clock0 source for Counter0 and Valid0 as XTAL, and Clock1 source for Counter1 as PLL output clock.
- Based on the equations defined in [Section 7.2.1](#), calculated seed values for Counters can be Counter0 = 29940; Valid0 = 120; Counter1 = 300000
- Once the DCC is enabled, the counters Counter0 and Counter1 both start counting down from the seed values.
- When Counter0 reaches zero, Counter0 automatically triggers the Valid0 counter.
- When Valid0 reaches zero and Counter1 is not zero, an ERROR status flag is set and a "DCC error" is sent to the PIE. Counter1 is frozen so that the counter stops counting down any further. The application can enable an interrupt to be generated from the PIE whenever this DCC error is indicated. Refer to the PIE Channel Mapping table in the *System Control and Interrupts* chapter to determine the channel mapping of the DCC Interrupt.
- The application then needs to clear the ERROR status flag and restart the DCC module so that the module is ready for the next spot measurement.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

### Error Conditions:

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0. This means that Clock1 is faster than expected, or Clock0 is slower than expected. This error includes the case when Clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that Clock1 is slower than expected. This error includes the case when Clock1 is stuck at 1 or 0.

Any error freezes the counters from counting. An application can then read out the counter values to help determine what caused the error.

### Example-2: Measuring AUXCLKIN frequency

Another example of single-shot mode is to measure the frequency of AUXCLKIN (unknown frequency) using INTOSC1 (10MHz) as the reference clock and SYSClk is 10MHz. The measurement sequence proceeds as follows:

- Set Clock0 source for Counter0 and Valid0 as INTOSC1 (10MHz), and Clock1 source for Counter1 as AUXCLKIN.
- Now configure counter values using equations in [Section 7.2.1](#). For tolerance = ±0.1%, Total Error = 10 clock0 cycles; Window = 10000 clock0 cycles; Counter0 = 9990; Valid0 = 20. Since Clock1 frequency (Fclk1) is unknown, the Counter1 value can be set to the maximum value, 1048575 (0xFFFFF).
- Once the DCC is enabled, the counters Counter0 and Counter1 both start counting down from the seed values.
- Since Counter1 is set to the maximum value, 1048575, the counter does not expire when Counter0 and Valid0 have expired. This generates an error that is expected and the application ignores this error and uses Counter1 values to compute the frequency of Clock1 (Fclk1).
- Knowing the frequency of Clock0 (INTOSC1), Fclk0 = 10MHz, and using [Equation 1](#), the frequency of AUXCLKIN, Fclk1, can be measured:

$$F_{clk1} = \frac{F_{clk0} \times (1048575 - Meas. Counter1)}{(Counter0 + Valid0)} = \frac{10 \times (1048575 - Meas. Counter1)}{(9990 + 20)} \quad (1)$$

### 7.2.3 Continuous Monitoring Mode

In this mode, the DCC is used by the application to make sure that two clock signals maintain the correct frequency ratio. Suppose the application wants to make sure that the PLL output signal always maintains a fixed frequency relationship with the XTAL:

- In this case, the application can use the XTAL as the Clock0 signal (for Counter0 and Valid0) and the PLL output as the Clock1 (for Counter1).
- The seed values of Counter0, Valid0 and Counter1 are selected based on the equations defined in [Section 7.2.1](#) such that if the actual frequencies of Clock0 and Clock1 are equal to the expected frequencies, then the Counter1 reaches zero during the count down of the Valid0 counter.
- If the Counter1 reaches zero during the count down of the Valid0 counter, then all the counters (Counter0, Valid0, Counter1) are reloaded with the initial seed values.
- This sequence of counting down and checking then continues as long as there is no error, or until the DCC module is disabled.
- The counters must get reloaded if the application resets and restarts the DCC module.

#### Error Conditions:

An error condition is generated by one of the following:

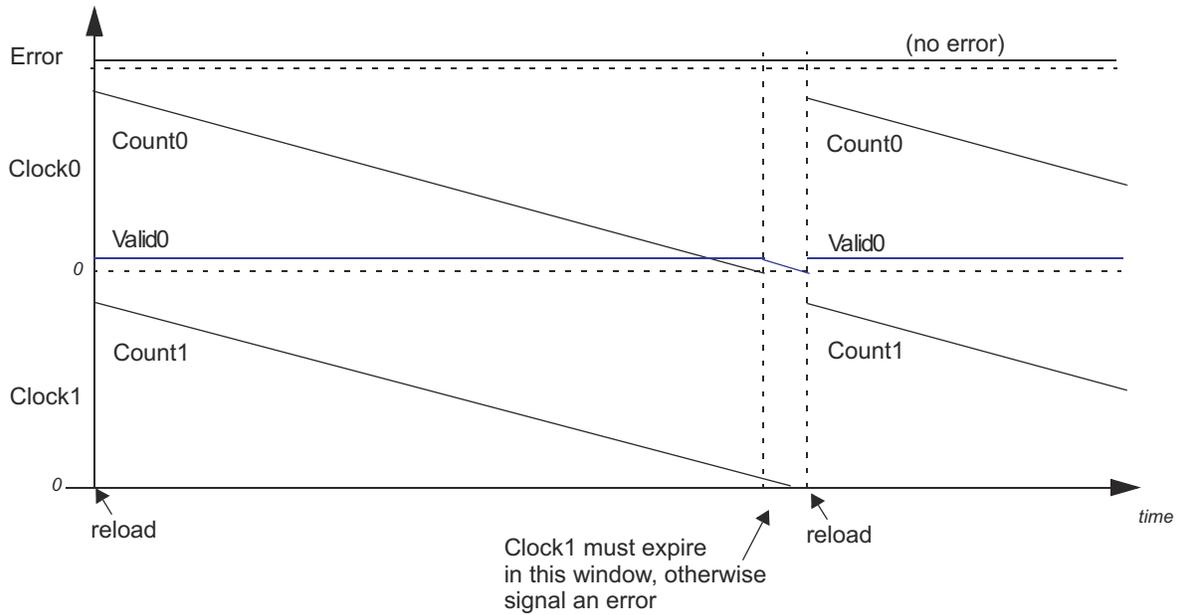
1. Counter1 counts down to 0 before Counter0 reaches 0. This means that Clock1 is faster than expected or Clock0 is slower than expected. This condition includes the case when Clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that Clock1 is slower than expected. This condition includes the case when Clock1 is stuck at 1 or 0.

Any error freezes the counters from counting. An application can then read out the counter values to help determine what caused the error.

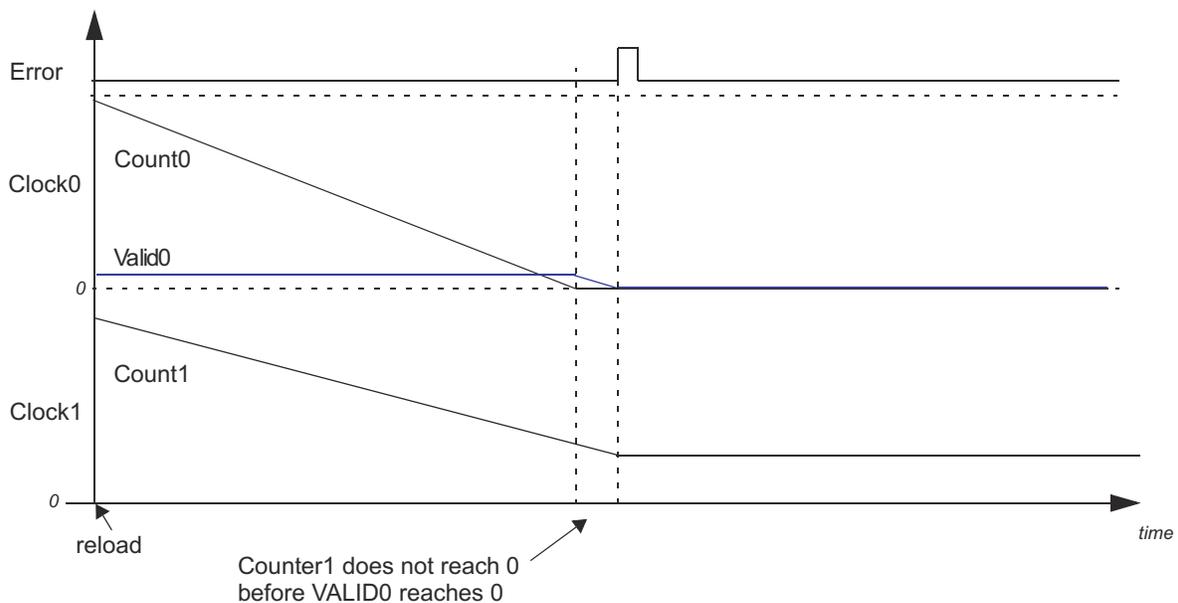
### 7.2.4 Error Conditions

While operating in continuous mode, the counters get reloaded with the seed values and continue counting down under the following conditions:

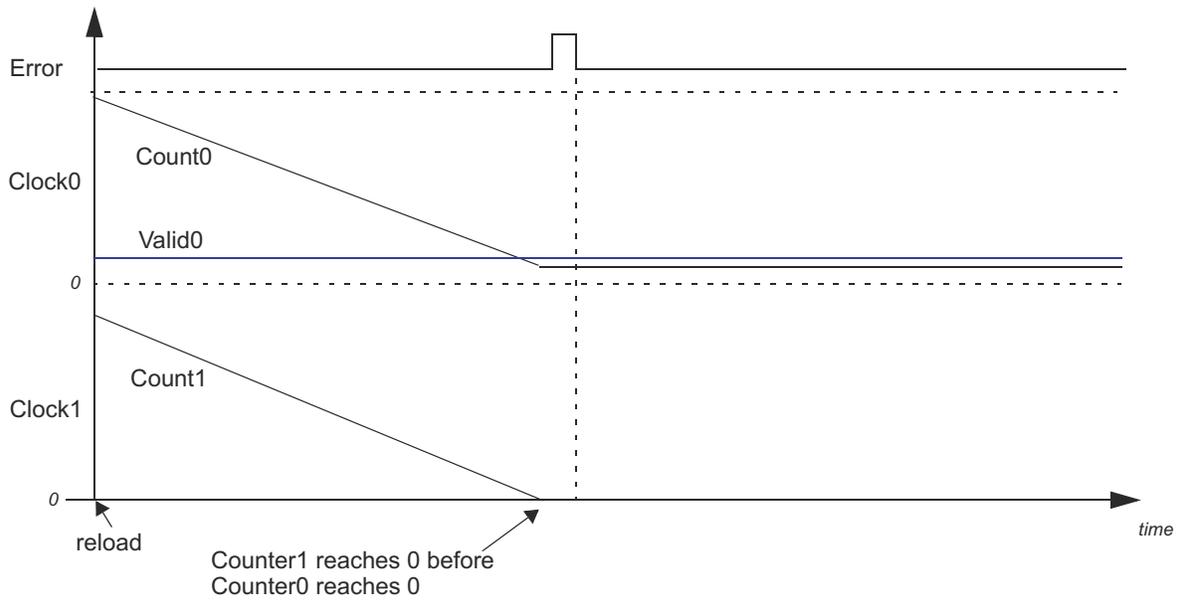
- The module is reset or restarted by the application, OR
- Counter0, Valid 0, and Counter1 all reach 0 without any error.



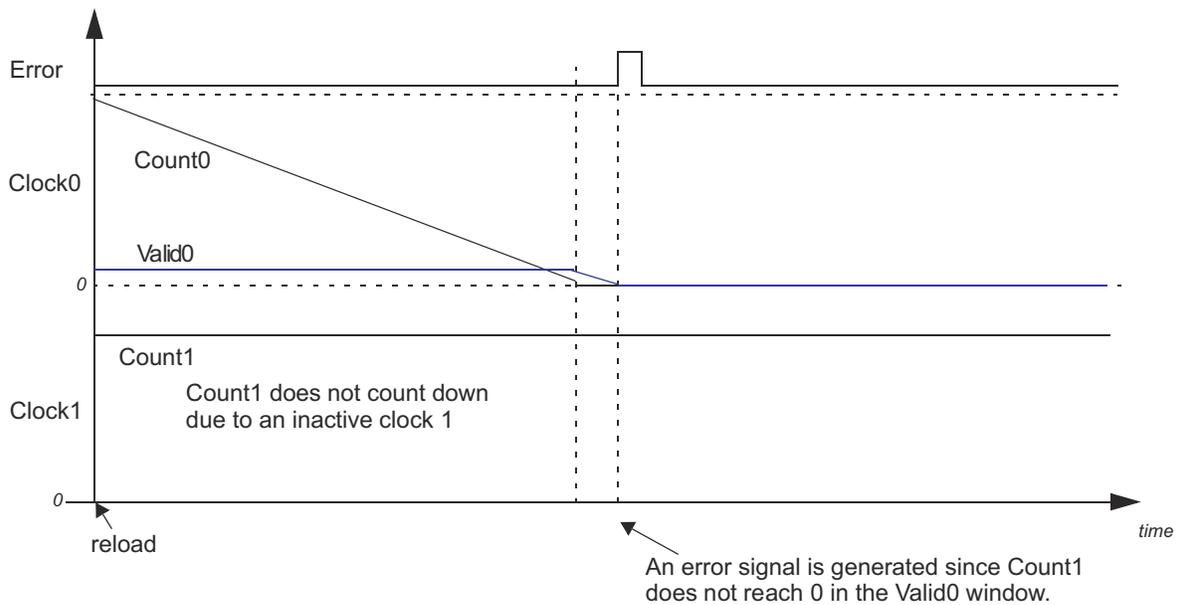
**Figure 7-3. Counter Relationship**



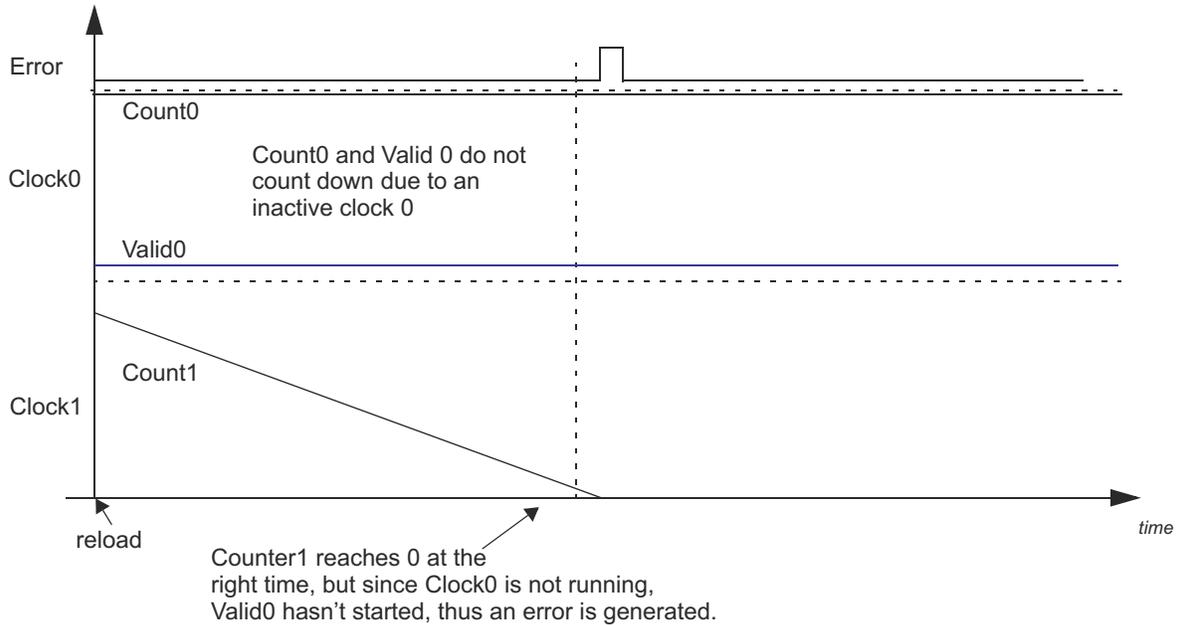
**Figure 7-4. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting**



**Figure 7-5. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting**



**Figure 7-6. Clock1 Not Present - Results in an Error and Stops Counting**



**Figure 7-7. Clock0 Not Present - Results in an Error and Stops Counting**

### 7.3 Interrupts

DCC generates an interrupt on either of two events:

- DCC finishes counting and all the counters expire within a defined window indicating DONE operation, provided `DCCGCTRL.DONENA=1`.
- DCC finishes counting with error where counters do not expire in a defined window. This indicates an ERROR event, and sets an interrupt provided `DCCGCTRL.ERRENA=1`.

Interrupts generated by DONE or ERROR events are ORed and flagged as a `SYS_ERR` interrupt. Refer to the PIE Channel Mapping table in the *System Control and Interrupts* chapter to determine the interrupt channel mapping. The application interrupt service routine needs to check the status flag inside the `DCCSTATUS` register to determine whether the interrupt is due to ERROR or DONE.

DCC Error interrupts can also be configured as a Non-Maskable Interrupt (NMI) by enabling the `CLKFAILCFG.DCCx_ERROR_EN` flag.

## 7.4 Software

### 7.4.1 DCC Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/dcc

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 7.4.1.1 DCC Single shot Clock verification

FILE: dcc\_ex1\_single\_shot\_verification.c

This program uses the XTAL clock as a reference clock to verify the frequency of the PLLRAW clock.

The Dual-Clock Comparator Module 0 is used for the clock verification. The clocksource0 is the reference clock (Fclk0 = 20Mhz) and the clocksource1 is the clock that needs to be verified (Fclk1 = 120Mhz). Seed is the value that gets loaded into the Counter.

Please refer to the TRM for details on counter seed values to be set.

NOTE: In this device, by default, the XTAL is disabled and INTOSC2 is configured as the PLL source. To use XTAL (of frequency 20MHz) as the PLL source, update the device.h file with following changes :

- Comment the line `#define USE_PLL_SRC_INTOSC`
- Uncomment the line `#define USE_PLL_SRC_XTAL` If you are using a XTAL with different frequency, update the macros `DEVICE_OSCSRC_FREQ`, `DEVICE_SETCLOCK_CFG`, `DEVICE_SYSClk_FREQ` accordingly.

To run this example, XTAL needs to be enabled. XTAL is enabled if you use it as the PLL source. You can also enable it by using the function `SysCtl_turnOnOsc(SYSCTL_OSCSRC_XTAL)`

#### External Connections

- None

#### Watch Variables

- *status/result* - Status of the PLLRAW clock verification

#### 7.4.1.2 DCC Single shot Clock measurement

FILE: dcc\_ex2\_single\_shot\_measurement.c

This program demonstrates Single Shot measurement of the INTOSC2 clock post trim using XTAL as the reference clock.

The Dual-Clock Comparator Module 0 is used for the clock measurement. The clocksource0 is the reference clock (Fclk0 = 20Mhz) and the clocksource1 is the clock that needs to be measured (Fclk1 = 10Mhz). Since the frequency of the clock1 needs to be measured an initial seed is set to the max value of the counter.

Please refer to the TRM for details on counter seed values to be set.

NOTE: In this device, by default, the XTAL is disabled and INTOSC2 is configured as the PLL source. To use XTAL (of frequency 20MHz) as the PLL source, update the device.h file with following changes :

- Comment the line `#define USE_PLL_SRC_INTOSC`
- Uncomment the line `#define USE_PLL_SRC_XTAL` If you are using a XTAL with different frequency, update the macros `DEVICE_OSCSRC_FREQ`, `DEVICE_SETCLOCK_CFG`, `DEVICE_SYSClk_FREQ` accordingly.

To run this example, XTAL needs to be enabled. XTAL is enabled if you use it as the PLL source. You can also enable it by using the function `SysCtl_turnOnOsc(SYSCTL_OSCSRC_XTAL)`

#### External Connections

- None

#### Watch Variables

- *result* - Status if the INTOSC2 clock measurement completed successfully.

- *meas\_freq1* - measured clock frequency, in this case for INTOSC2.

#### 7.4.1.3 DCC Continuous clock monitoring

FILE: dcc\_ex3\_continuous\_monitoring\_of\_clock.c

This program demonstrates continuous monitoring of PLL Clock in the system using INTOSC2 as the reference clock. This would trigger an interrupt on any error, causing the decrement/ reload of counters to stop.

The Dual-Clock Comparator Module 0 is used for the clock monitoring. The clocksource0 is the reference clock (Fclk0 = 10Mhz) and the clocksource1 is the clock that needs to be monitored (Fclk1 = 100Mhz). The clock0 and clock1 seed are set to achieve a window of 300us. Seed is the value that gets loaded into the Counter. For the sake of demo a slight variance is given to clock1 seed value to generate an error on continuous monitoring.

Please refer to the TRM for details on counter seed values to be set. Note : When running in flash configuration it is good to do a reset & restart after loading the example to remove any stale flags/states.

##### External Connections

- None

##### Watch Variables

- *status/result* - Status of the PLLRAW clock monitoring
- *cnt0* - Counter0 Value measure when error is generated
- *cnt1* - Counter1 Value measure when error is generated
- *valid* - Valid0 Value measure when error is generated

#### 7.4.1.4 DCC Continuous clock monitoring

FILE: dcc\_ex3\_continuous\_monitoring\_of\_clock\_syscfg.c

This program demonstrates continuous monitoring of PLL Clock in the system using INTOSC2 as the reference clock. This would trigger an interrupt on any error, causing the decrement/ reload of counters to stop. The Dual-Clock Comparator Module 0 is used for the clock monitoring. The clocksource0 is the reference clock (Fclk0 = 10Mhz) and the clocksource1 is the clock that needs to be monitored (Fclk1 = 100Mhz). The clock0 and clock1 seed are set automatically by the error tolerances defined in the sysconfig file included this project. For the sake of demo an un-realistic tolerance is assumed to generate an error on continuous monitoring.

Please refer to the TRM for details on counter seed values to be set. Note : When running in flash configuration it is good to do a reset & restart after loading the example to remove any stale flags/states.

##### External Connections

- None

##### Watch Variables

- *status/result* - Status of the PLLRAW clock monitoring
- *cnt0* - Counter0 Value measure when error is generated
- *cnt1* - Counter1 Value measure when error is generated
- *valid* - Valid0 Value measure when error is generated

#### 7.4.1.5 DCC Detection of clock failure

FILE: dcc\_ex4\_clock\_fail\_detect.c

This program demonstrates clock failure detection on continuous monitoring of the PLL Clock in the system using XTAL as the osc clock source. Once the oscillator clock fails, it would trigger a DCC error interrupt, causing the decrement/ reload of counters to stop. In this examples, the clock failure is simulated by turning off the XTAL oscillator. Once the ISR is serviced, the osc source is changed to INTOSC1 and the PLL is turned off.

The Dual-Clock Comparator Module 0 is used for the clock monitoring. The clocksource0 is the reference clock (Fclk0 = 20Mhz) and the clocksource1 is the clock that needs to be monitored (Fclk1 = 100Mhz). Seed is the value that gets loaded into the Counter.

In the current example, the XTAL is expected to be a Resonator running in Crystal mode which is later switched off to simulate the clock failure. If an SE Crystal is used, you will need to physically disconnect the clock on the board. Please refer to the TRM for details on counter seed values to be set. Note : When running in flash configuration it is good to do a reset & restart after loading the example to remove any stale flags/states.

NOTE: In this device, by default, the XTAL is disabled and INTOSC2 is configured as the PLL source. To use XTAL (of frequency 20MHz) as the PLL source, update the device.h file with following changes :

- Comment the line `#define USE_PLL_SRC_INTOSC`
- Uncomment the line `#define USE_PLL_SRC_XTAL` If you are using a XTAL with different frequency, update the macros `DEVICE_OSCSRC_FREQ`, `DEVICE_SETCLOCK_CFG`, `DEVICE_SYSCLK_FREQ` accordingly.

To run this example, XTAL needs to be configured as PLL source.

#### External Connections

- None

#### Watch Variables

- `status/result` - Status of the clock failure detection

## 7.5 DCC Registers

This section describes the Dual Clock Comparator Registers.

#### Note

DCC is used by Boot ROM, hence the register values are different than hardware reset value. Make sure to configure the values of these registers to the desired value before enabling DCC.

### 7.5.1 DCC Base Address Table

**Table 7-1. DCC Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
Dcc0Regs	DCC_REGS	DCC0_BASE	0x0005_E700	YES

## 7.5.2 DCC\_REGS Registers

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

**Table 7-2. DCC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	DCCGCTRL	Starts / stops the counters. Clears the error signal.		<a href="#">Go</a>
8h	DCCCNTSEED0	Seed value for the counter attached to Clock Source 0.		<a href="#">Go</a>
Ch	DCCVALIDSEED0	Seed value for the timeout counter attached to Clock Source 0.		<a href="#">Go</a>
10h	DCCCNTSEED1	Seed value for the counter attached to Clock Source 1.		<a href="#">Go</a>
14h	DCCSTATUS	Specifies the status of the DCC Module.		<a href="#">Go</a>
18h	DCCCNT0	Value of the counter attached to Clock Source 0.		<a href="#">Go</a>
1Ch	DCCVALID0	Value of the valid counter attached to Clock Source 0.		<a href="#">Go</a>
20h	DCCCNT1	Value of the counter attached to Clock Source 1.		<a href="#">Go</a>
24h	DCCCLKSRC1	Selects the clock source for Counter 1.		<a href="#">Go</a>
28h	DCCCLKSRC0	Selects the clock source for Counter 0.		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 7-3 shows the codes that are used for access types in this section.

**Table 7-3. DCC\_REGS Access Type Codes**

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

### 7.5.2.1 DCCGCTRL Register (Offset = 0h) [Reset = 00005555h]

DCCGCTRL is shown in [Figure 7-8](#) and described in [Table 7-4](#).

Return to the [Summary Table](#).

Starts / stops the counters. Clears the error signal.

**Figure 7-8. DCCGCTRL 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
DONEENA				SINGLESHOT				ERRENA				DCCENA			
R/W-5h				R/W-5h				R/W-5h				R/W-5h			

**Table 7-4. DCCGCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	DONEENA	R/W	5h	DONE Enable Enables/disables the done interrupt signal, but has no effect on the done status flag in DCCSTAT register. 0101 The done signal is disabled Others The done signal is enabled Reset type: SYSRSn
11-8	SINGLESHOT	R/W	5h	Single-Shot Enable Enables/disables repetitive operation of the DCC. 1010: Stop counting when COUNTER0 and VALID0 both reach zero 1011: Reserved Others: Continuously repeat (until error) Reset type: SYSRSn
7-4	ERRENA	R/W	5h	Error Enable Enables/disables the error signal. 0101 The error signal is disabled Others The error signal is enabled Reset type: SYSRSn
3-0	DCCENA	R/W	5h	DCC Enable Starts and stops the operation of the DCC. 0101 Counters are stopped Others Counters are running Reset type: SYSRSn

### 7.5.2.2 DCCNTSEED0 Register (Offset = 8h) [Reset = 0000000h]

DCCNTSEED0 is shown in [Figure 7-9](#) and described in [Table 7-5](#).

Return to the [Summary Table](#).

Seed value for the counter attached to Clock Source 0.

**Figure 7-9. DCCNTSEED0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNTSEED0																			
R-0h												R/W-0h																			

**Table 7-5. DCCNTSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNTSEED0	R/W	0h	Seed Value for Counter 0 Contains the seed value that gets loaded into Counter 0 (Clock Source 0). NOTE: Operating the DCC with '0' in the COUNTSEED0 register will result in undefined operation. Reset type: SYSRSn

### 7.5.2.3 DCCVALIDSEED0 Register (Offset = Ch) [Reset = 0000000h]

DCCVALIDSEED0 is shown in [Figure 7-10](#) and described in [Table 7-6](#).

Return to the [Summary Table](#).

Seed value for the timeout counter attached to Clock Source 0.

**Figure 7-10. DCCVALIDSEED0 Register**

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

**Table 7-6. DCCVALIDSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALIDSEED	R/W	0h	Seed Value for Valid Duration Counter 0 Contains the seed value that gets loaded into the valid duration counter for Clock Source 0. NOTE: Operating the DCC with '0' in the VALIDSEED0 register will result in undefined operation. VALID0 defines a window in which COUNT1 expires. This window is meant to be at least four cycles wide. Do not program a value less than '4' into the VALID0 register. Reset type: SYSRSn

### 7.5.2.4 DCCNTSEED1 Register (Offset = 10h) [Reset = 0000000h]

DCCNTSEED1 is shown in [Figure 7-11](#) and described in [Table 7-7](#).

Return to the [Summary Table](#).

Seed value for the counter attached to Clock Source 1.

**Figure 7-11. DCCNTSEED1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNTSEED1																			
R-0h												R/W-0h																			

**Table 7-7. DCCNTSEED1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNTSEED1	R/W	0h	Seed Value for Counter 1 Contains the seed value that gets loaded into Counter 1 (Clock Source 1). NOTE: Operating the DCC with '0' in the COUNTSEED1 register will result in undefined operation. Reset type: SYSRSn

### 7.5.2.5 DCCSTATUS Register (Offset = 14h) [Reset = 0000000h]

DCCSTATUS is shown in [Figure 7-12](#) and described in [Table 7-8](#).

Return to the [Summary Table](#).

Specifies the status of the DCC Module.

**Figure 7-12. DCCSTATUS Register**

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

**Table 7-8. DCCSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	DONE	R/W	0h	Single-Shot Done Flag Indicates when single-shot mode is complete without error. Writing a '1' to this bit clears the flag. 0 Single-shot mode has not completed. 1 Single-shot mode has completed. Reset type: SYSRSn
0	ERR	R/W	0h	Error Flag Indicates whether or not an error has occurred. Writing a '1' to this bit clears the flag. 0 No errors have occurred. 1 An error has occurred. Reset type: SYSRSn

### 7.5.2.6 DCCNT0 Register (Offset = 18h) [Reset = 0000000h]

DCCNT0 is shown in [Figure 7-13](#) and described in [Table 7-9](#).

Return to the [Summary Table](#).

Value of the counter attached to Clock Source 0.

**Figure 7-13. DCCNT0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNT0																			
R-0h												R-0h																			

**Table 7-9. DCCNT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNT0	R	0h	Current Value of Counter 0 Reset type: SYSRSn

### 7.5.2.7 DCCVALID0 Register (Offset = 1Ch) [Reset = 00000000h]

DCCVALID0 is shown in [Figure 7-14](#) and described in [Table 7-10](#).

Return to the [Summary Table](#).

Value of the valid counter attached to Clock Source 0.

**Figure 7-14. DCCVALID0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VALID0															
R-0h																R-0h															

**Table 7-10. DCCVALID0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VALID0	R	0h	Current Value of Valid 0 Reset type: SYSRSn

### 7.5.2.8 DCCNT1 Register (Offset = 20h) [Reset = 0000000h]

DCCNT1 is shown in [Figure 7-15](#) and described in [Table 7-11](#).

Return to the [Summary Table](#).

Value of the counter attached to Clock Source 1.

**Figure 7-15. DCCNT1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNT1																			
R-0h												R-0h																			

**Table 7-11. DCCNT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-0	COUNT1	R	0h	Current Value of Counter 1 Reset type: SYSRSn

### 7.5.2.9 DCCCLKSRC1 Register (Offset = 24h) [Reset = 0000000h]

DCCCLKSRC1 is shown in [Figure 7-16](#) and described in [Table 7-12](#).

Return to the [Summary Table](#).

Selects the clock source for Counter 1.

**Figure 7-16. DCCCLKSRC1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY				RESERVED						CLKSRC1					
R-0/W-0h				R-0h						R/W-0h					

**Table 7-12. DCCCLKSRC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	KEY	R-0/W	0h	Enables or Disables Clock Source Write for COUNT1 1010 The CLKSRC field selects the clock source for COUNT1. Others: Previous values retained new writes on register fields has no impact. Reset type: SYSRSn
11-6	RESERVED	R	0h	Reserved

**Table 7-12. DCCCLKSRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-0	CLKSRC1	R/W	0h	<p>Clock Source Select for Counter 1 Specifies the clock source for COUNT1, when the KEY field enables this feature. Note: Any values not explicitly defined below are reserved. Reset type: SYSRSn</p> <p>0h (R/W) = Direct output of SYSPLL CLKOUT 1h (R/W) = Reserved 2h (R/W) = INTOSC1 output clock 3h (R/W) = INTOSC2 output clock 4h (R/W) = Reserved 5h (R/W) = Reserved 6h (R/W) = System clock 7h (R/W) = Reserved 8h (R/W) = Reserved 9h (R/W) = Input 15 of INPUTXBAR1 Ah (R/W) = Auxiliary clock input Bh (R/W) = Clock input to EPWM module Ch (R/W) = Bit clock for SPI and SCI modules Dh (R/W) = ADC conversion clock Eh (R/W) = Watchdog clock after dividers Fh (R/W) = CAN0 bit clock 10h (R/W) = Reserved 11h (R/W) = Reserved 12h (R/W) = Reserved 13h (R/W) = Reserved 14h (R/W) = Reserved 15h (R/W) = Reserved 16h (R/W) = Reserved 17h (R/W) = FCLK (divided clock) output from Flash wrapper 18h (R/W) = Input 11 of INPUTXBAR1 19h (R/W) = Input 12 of INPUTXBAR1 1Ah (R/W) = Reserved 1Bh (R/W) = Reserved 1Ch (R/W) = Reserved 1Dh (R/W) = Reserved 1Eh (R/W) = Reserved 1Fh (R/W) = Reserved</p>

### 7.5.2.10 DCCCLKSRC0 Register (Offset = 28h) [Reset = 0000000h]

DCCCLKSRC0 is shown in [Figure 7-17](#) and described in [Table 7-13](#).

Return to the [Summary Table](#).

Selects the clock source for Counter 0.

**Figure 7-17. DCCCLKSRC0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY				RESERVED								CLKSRC0			
R-0/W-0h				R-0h								R/W-0h			

**Table 7-13. DCCCLKSRC0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-12	KEY	R-0/W	0h	Enables or Disables Clock Source Write for COUNT0 1010: The CLKSRC0 field written with key gets updated to with new selection to clock COUNT0. Others: Previous values retained new writes on register fields has no impact. Reset type: SYSRSn
11-5	RESERVED	R	0h	Reserved
4-0	CLKSRC0	R/W	0h	Clock Source Select for Counter 0 Specifies the clock source for COUNT0, when the KEY field enables this feature. Note: All values not defined below are reserved. Reset type: SYSRSn 0h (R/W) = Crystal oscillator output 1h (R/W) = INTOSC1 output 2h (R/W) = INTOSC2 output 4h (R/W) = TCK pin input 5h (R/W) = CPU1 system clock 8h (R/W) = Auxiliary clock input Ch (R/W) = Input 16 of INPUTXBAR1 Eh (R/W) = Reserved Fh (R/W) = Reserved

### 7.5.3 DCC Registers to Driverlib Functions

**Table 7-14. DCC Registers to Driverlib Functions**

File	Driverlib Function
<b>DCCGCTRL</b>	
dcc.h	DCC_enableModule
dcc.h	DCC_disableModule
dcc.h	DCC_enableErrorSignal
dcc.h	DCC_enableDoneSignal
dcc.h	DCC_disableErrorSignal
dcc.h	DCC_disableDoneSignal
dcc.h	DCC_enableSingleShotMode
dcc.h	DCC_disableSingleShotMode
<b>DCCNTSEED0</b>	
dcc.h	DCC_setCounterSeeds

**Table 7-14. DCC Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>DCCVALIDSEED0</b>	
dcc.h	DCC_setCounterSeeds
<b>DCCCNTSEED1</b>	
dcc.h	DCC_setCounterSeeds
<b>DCCSTATUS</b>	
dcc.h	DCC_getErrorStatus
dcc.h	DCC_getSingleShotStatus
dcc.h	DCC_clearErrorFlag
dcc.h	DCC_clearDoneFlag
sysctl.c	SysCtl_isPLLValid
<b>DCCCNT0</b>	
dcc.h	DCC_getCounter0Value
<b>DCCVALID0</b>	
dcc.h	DCC_getValidCounter0Value
<b>DCCCNT1</b>	
dcc.h	DCC_getCounter1Value
<b>DCCCLKSRC1</b>	
dcc.h	DCC_setCounter1ClkSource
dcc.h	DCC_getCounter1ClkSource
<b>DCCCLKSRC0</b>	
dcc.h	DCC_setCounter0ClkSource
dcc.h	DCC_getCounter0ClkSource

This page intentionally left blank.

**General-Purpose Input/Output (GPIO)**

The GPIO module controls the device's digital and analog I/O multiplexing, which uses shared pins to maximize application flexibility. The pins are named by the general-purpose I/O name (for example, GPIO0, GPIO25, GPIO58). These pins can be individually selected to operate as digital I/O (also called GPIO mode), or connected to one of several peripheral I/O signals. The input signals can be qualified to remove unwanted noise.

<b>8.1 Introduction</b> .....	<b>602</b>
<b>8.2 Configuration Overview</b> .....	<b>604</b>
<b>8.3 Digital Inputs on ADC Pins (AIOs)</b> .....	<b>605</b>
<b>8.4 Digital Inputs and Outputs on ADC Pins (AGPIOs)</b> .....	<b>605</b>
<b>8.5 Digital General-Purpose I/O Control</b> .....	<b>606</b>
<b>8.6 Input Qualification</b> .....	<b>607</b>
<b>8.7 GPIO and Peripheral Muxing</b> .....	<b>611</b>
<b>8.8 Internal Pullup Configuration Requirements</b> .....	<b>614</b>
<b>8.9 Software</b> .....	<b>615</b>
<b>8.10 GPIO Registers</b> .....	<b>616</b>

## 8.1 Introduction

Up to twelve independent peripheral signals are multiplexed on a single GPIO-enabled pin in addition to the CPU-controlled I/O capability. Each pin output can be controlled by either a peripheral or one of the CPU masters.

- CPU1

There are up to 8 possible 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-GPIO191
- Port G consists of GPIO192-GPIO223
- Port H consists of GPIO224-GPIO255

---

### Note

Some GPIO and I/O ports can be unavailable on particular devices. See the *GPIO Registers* section for available GPIO and I/O ports.

---

The analog signals on this device are multiplexed with digital inputs and outputs. Some of these analog IO (AIO) pins do not have digital output capability. Others of these pins are analog pins capable of full digital input and output capability (AGPIO). Analog pins with AIO (digital input only) capability contain "AIO" signals in the Pin Attributes table of the device data sheet. Analog pins with full input and output capability (AGPIO pins) contain "GPIO" signals in the Pin Attributes table of the device data sheet. AGPIO pins also have pin names with both analog signals and GPIO in the name.

[Figure 8-1](#) shows the GPIO logic for a single pin.

There are two key features to note in [Figure 8-1](#). 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, for the CPU to read the physical state of the pin independent of peripheral muxing is possible. Likewise, external interrupts can be generated from peripheral activity. All pin options such as input qualification and open-drain output are valid for all peripherals.

---

### Note

JTAG uses a different signal path that does not support inversion or qualification.

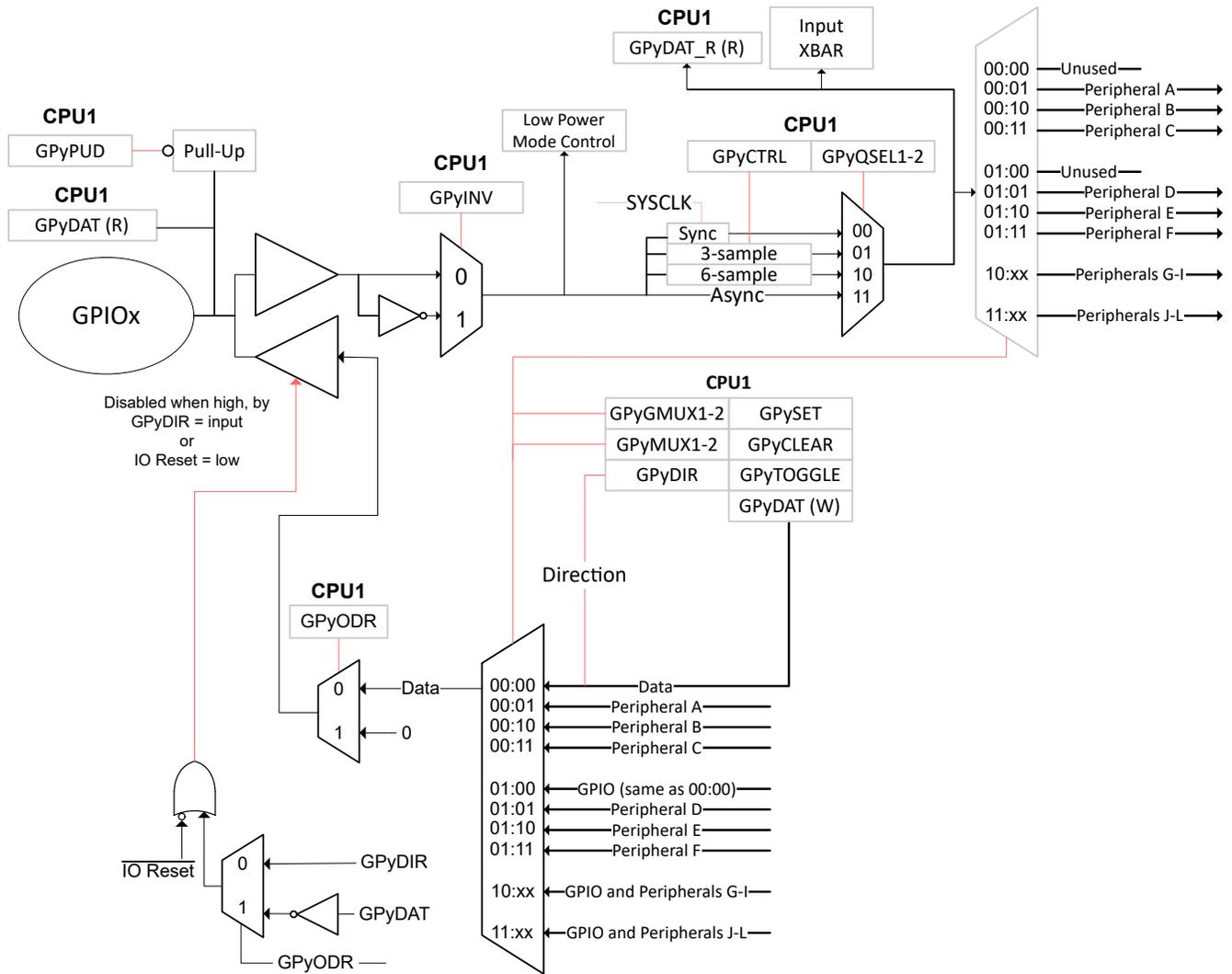
GPIO18/X2 and GPIO19/X1 have different timings due to the load placed on them by the oscillator circuit. For information on using GPIO18/X2 and GPIO19/X1 as GPIOs, see the data sheet and the *Clocking* section of this document.

If digital signals with sharp edges (high dv/dt) are connected to the AIOs or AGPIOs, cross-talk can occur with adjacent analog signals. Therefore, limit the edge rate of signals connected to AIOs or AGPIOs if adjacent channels are being used for analog functions.

---

**Note**

In open-drain mode, the GPIO does not drive the pin high, the GPIO can only pull the pin low. Instead, use an external pull-up to the bus voltage to drive the high level. When open-drain mode is enabled, the value in the GPyDAT register still controls the pin state. Writing a value of 1 turns off the driver to allow the external pull-up to control the pin; writing a value of 0 pulls the pin to ground. The open-drain configuration is automatically used by peripherals such as I2C and PMBus (no need to enable open-drain mode locally). This mode can also be set manually by writing to the GPyODR register and can be used when there are multiple nodes on the same net to avoid the pin contention that a push-pull driver can cause.



**Figure 8-1. GPIO Logic for a Single Pin**

## 8.1.1 GPIO Related Collateral

### Foundational Materials

- [C2000 Academy - GPIO](#)

### Getting Started Materials

- [How to Maximize GPIO Usage in C2000 Devices Application Report](#)
- [\[FAQ\] C2000 GPIO FAQ](#)

## 8.2 Configuration Overview

I/O pin configuration consists of several steps:

1. **Plan the device pin-out:** Make a list of all required peripherals for the application. Using the peripheral mux information in the device data sheet, choose which GPIOs to use for the peripheral signals. Decide which of the remaining GPIOs to use as inputs and outputs.

Once the peripheral muxing has been chosen, implement the mux 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, with the exception of GPIO35 and GPIO37.

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 is 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, with the exception of GPIO35 and GPIO37. For an explanation of input qualification, see [Section 8.6](#).
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.

The GPyDAT\_R register can be used to read what value was written to the GPyDAT register.

5. **Select low-power mode wake-up sources:** GPIOs 0-63 can be used to wake the system up from low power modes. To select one or more GPIOs for wake-up, write to the appropriate bits in the GPIOLPMSEL0 and GPIOLPMSEL1 registers. These registers are part of the CPU system register space. For more information on low-power modes and GPIO wake-up, see the Low-Power Modes section in the *System Control and Interrupts* chapter.
6. **Select external interrupt sources:** Configuring external interrupts is a two-step process. First, the interrupts themselves must be enabled and the polarity must be configured using the XINTnCR registers. Second, the XINT1-5 GPIO pins must be set by selecting the sources for Input X-BAR signals 4, 5, 6, 13, and 14, respectively. For more information on the Input X-BAR architecture, see the *Crossbar (X-BAR)* chapter.

### 8.3 Digital Inputs on ADC Pins (AIOs)

Some GPIOs are multiplexed with analog pins and only have digital input functionality. These are also referred to as AIOs. Pins with only an AIO option on this port can only function in input mode. See the device data sheet for list of AIO signals. By default, these pins function as analog pins and the GPIOs are in a high-impedance state. The GPyAMSEL register is used to configure these pins for digital or analog operation.

---

#### Note

If digital signals with sharp edges (high dv/dt) are connected to the AIOs, cross-talk can occur with adjacent analog signals. Therefore, limit the edge rate of signals connected to AIOs if adjacent channels are being used for analog functions.

---

### 8.4 Digital Inputs and Outputs on ADC Pins (AGPIOs)

Some GPIOs are multiplexed with analog pins and have digital input and output functionality. These are also referred to as AGPIOs. Unlike AIOs, AGPIOs have full input and output capability.

By default, the AGPIOs are not connected and must be configured. [Table 8-1](#) shows how to configure the AGPIOs. To enable the analog functionality, set the register AGPICTRLx from analog subsystem. To enable the digital functionality, set the register GPxAMSEL from the *General-Purpose Input/Output (GPIO)* chapter.

**Table 8-1. AGPIO Configuration**

AGPICTRLx.GPIOy (Default = 0)	GPxAMSEL.GPIOy (Default = 1)	Pin Connected To:	
		ADC	GPIOy
0	0	-	Yes
<b>0</b>	<b>1</b>	- <sup>(1)</sup>	- <sup>(1)</sup>
1	0	-	Yes
1	1	Yes	-

(1) By default there are no signals connected to AGPIO pins. One of the other rows in the table must be chosen for pin functionality.

---

#### Note

If digital signals with sharp edges (high dv/dt) are connected to the AGPIOs, cross-talk can occur with adjacent analog signals. The user must therefore limit the edge rate of signals connected to AGPIOs, if adjacent channels are being used for analog functions.

---

## 8.5 Digital General-Purpose I/O Control

The values on the pins that are configured as GPIO can be changed by using the following 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 is also driven either low or high. If the pin is not configured as a GPIO output, then the value is latched but the pin is not driven. Only if the pin is later configured as a GPIO output is the latched value driven onto the pin.

When using the GPyDAT register to change the level of an output pin, be cautious to not accidentally change the level of another pin. For example, to change the output latch level of GPIOA0 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 can 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) reads the old value and writes the value 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 waits for the first to finish the write due to the write-followed-by-read protection on this peripheral frame. There is 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 reads the old value of GPIO1 (0) and writes the value back along with the new value of GPIO2 (1). Therefore, GPIO1 pin stays low.

One answer is to put some NOPs between instructions. A better answer 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 bits that are currently in the process of changing.

- **GPyDAT\_R Registers**

The GPyDAT\_R registers are read only registers that return the value written to the GPyDAT registers instead of pin status. Writes to these registers have no effect.

- **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 sets the output latch high and the corresponding pin is driven high. If the pin is not configured as a GPIO output, then the value is latched but the pin is not driven. Only if the pin is later configured as a GPIO output is the latched value 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 clears the output latch and the pin is driven low. If the pin is not configured as a GPIO output, then the value is latched but the pin is not driven. Only if the pin is later configured as a GPIO output is the latched value 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 pulls the pin high. Likewise, if the output pin is high, then writing a 1 to the corresponding bit in the toggle register pulls the pin low. If the pin is not configured as a GPIO output, then the value is latched but the pin is not driven. Only if the pin is later configured as a GPIO output is the latched value driven onto the pin. Writing a 0 to any bit in the toggle registers has no effect.

## 8.6 Input Qualification

The input qualification scheme has been designed to be very flexible. 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 synchronized to SYSCLKOUT or qualification by a sampling window. For pins that are configured as peripheral inputs, the input can also be asynchronous in addition to synchronized to SYSCLKOUT or qualified by a sampling window. The remainder of this section describes the options available.

### 8.6.1 No Synchronization (Asynchronous Input)

This mode is used for peripherals where input synchronization is not required or the peripheral performs the synchronization. Examples include communication ports McBSP, SCI, SPI, and I<sup>2</sup>C. In addition, the ePWM trip zone ( $\overline{TZn}$ ) signals can function independent of the presence of SYSCLKOUT.

---

#### Note

Using input synchronization when the peripheral performs the synchronization can cause unexpected results. The user must make sure that the GPIO pin is configured for asynchronous in this case.

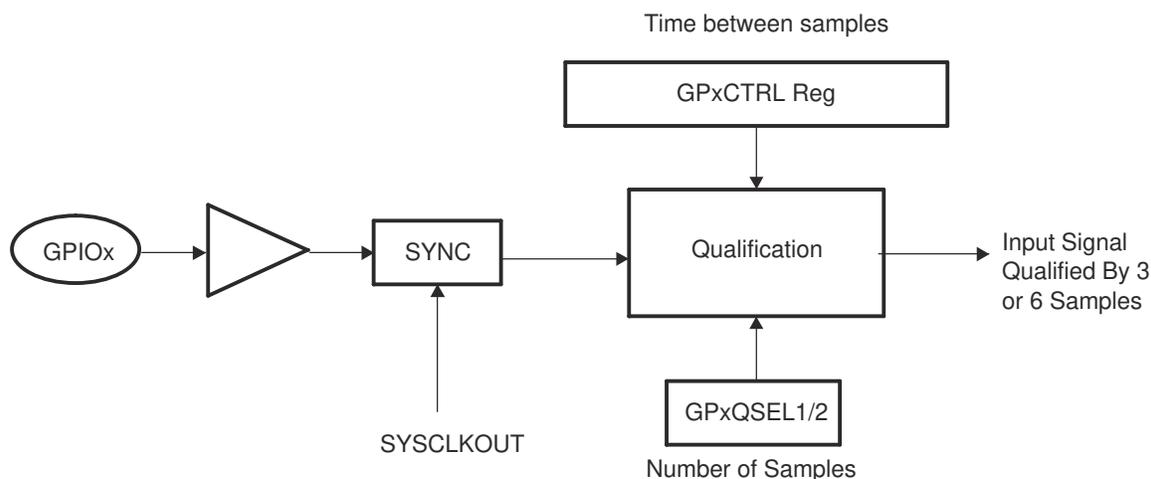
---

### 8.6.2 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, a SYSCLKOUT period of delay is needed for the input to the device to be changed. No further qualification is performed on the signal.

### 8.6.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLKOUT) and then qualified by a specified number of cycles before the input is allowed to change. Figure 8-2 and Figure 8-3 show how the input qualification is performed to eliminate unwanted noise. Two parameters are specified by the user for this type of qualification: 1) the sampling period, or how often the signal is sampled, and 2) the number of samples to be taken.



**Figure 8-2. Input Qualification Using a Sampling Window**

#### Time between samples (sampling period):

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal is 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 GPxCTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPxCTRL[QUALPRD1]. Table 8-2 and Table 8-3 show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

**Table 8-2. Sampling Period**

Sampling Period	
If GPxCTRL[QUALPRDn] = 0	$1 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] ≠ 0	$2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT	

**Table 8-3. Sampling Frequency**

Sampling Frequency	
If GPxCTRL[QUALPRDn] = 0	$f_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] ≠ 0	$f_{\text{SYSCLKOUT}} \times 1 \div (2 \times \text{GPxCTRL[QUALPRDn]})$
Where $f_{\text{SYSCLKOUT}}$ is the frequency of SYSCLKOUT	

From these equations, the minimum and maximum time between samples can be calculated for a given SYSCLKOUT frequency:

**Example: Maximum Sampling Frequency:**

If GPxCTRL[QUALPRDn] = 0

then the sampling frequency is  $f_{\text{SYSCLKOUT}}$

If, for example,  $f_{\text{SYSCLKOUT}} = 60\text{MHz}$

then the signal is sampled at 60MHz or one sample every 16.67ns.

**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 is sampled at  $60\text{MHz} \times 1 \div (2 \times 255)$  (117.647kHz) or one sample every 8.5 $\mu\text{s}$ .

**Number of samples:**

The number of times the signal is sampled is either three samples or six samples as specified in the qualification selection (GPyQSEL1, GPyQSEL2) registers. When three or six consecutive cycles are the same, then the input change is passed through to the device.

**Total Sampling-Window Width:**

The sampling window is the time during which the input signal is sampled as shown in [Figure 8-3](#). By using the equation for the sampling period, along with the number of samples to be taken, the total width of the window can be determined.

For the input qualifier to detect a change in the input, the level of the signal must be stable for the duration of the sampling-window width or longer.

The number of sampling periods within the window is always one less than the number of samples taken. For a three-sample window, the sampling-window width is two sampling-periods wide where the sampling period is defined in [Table 8-2](#). Likewise, for a six-sample window, the sampling-window width is five sampling-periods wide. [Table 8-4](#) and [Table 8-5](#) show the calculations used to determine the total sampling-window width based on GPxCTRL[QUALPRDn] and the number of samples taken.

**Table 8-4. Case 1: Three-Sample Sampling-Window Width**

Total Sampling-Window Width	
If GPxCTRL[QUALPRDn] = 0	$2 \times T_{\text{SYSCLKOUT}}$
If GPxCTRL[QUALPRDn] $\neq$ 0	$2 \times 2 \times \text{GPxCTRL[QUALPRDn]} \times T_{\text{SYSCLKOUT}}$
Where $T_{\text{SYSCLKOUT}}$ is the period in time of SYSCLKOUT	

**Table 8-5. 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 must 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 data sheet.

### Example Qualification Window:

For the example shown in Figure 8-3, the input qualification has been configured as follows:

- $\text{GPxQSEL1/2} = 1,0$ . This indicates a six-sample qualification.
- $\text{GPxCTRL}[\text{QUALPRDn}] = 1$ . The sampling period is  $t_w(\text{SP}) = 2 \times \text{GPxCTRL}[\text{QUALPRDn}] \times T_{\text{SYSCLKOUT}} = 2 \times T_{\text{SYSCLKOUT}}$ .

This configuration results in the following:

- The width of the sampling window is:

$$t_w(\text{IQSW}) = 5 \times t_w(\text{SP}) = 5 \times 2 \times \text{GPxCTRL}[\text{QUALPRDn}] \times T_{\text{SYSCLKOUT}} = 5 \times 2 \times T_{\text{SYSCLKOUT}}$$

- If, for example,  $T_{\text{SYSCLKOUT}} = 16.67\text{ns}$ , then the duration of the sampling window is:

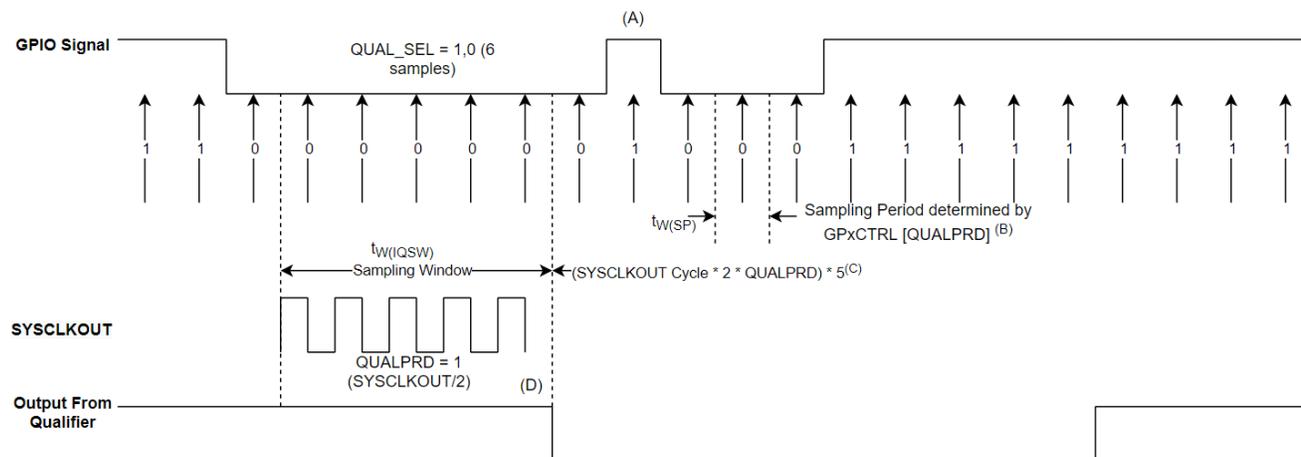
$$\text{Sampling period, } t_w(\text{SP}) = 2 \times T_{\text{SYSCLKOUT}} = 2 \times 16.67\text{ns} = 33.3\text{ns}$$

$$\text{Sampling window, } t_w(\text{IQSW}) = 5 \times t_w(\text{SP}) = 5 \times 33.3\text{ns} = 166.7\text{ns}$$

- To account for the asynchronous nature of the input relative to the sampling period and SYSCLKOUT, up to a single additional sampling period and SYSCLK period is required to detect a change in the input signal. For this example:

$$t_w(\text{IQSW}) + t_w(\text{SP}) + T_{\text{SYSCLKOUT}} = 166.7\text{ns} + 33.3\text{ns} + 16.67\text{ns} = 216.7\text{ns}$$

- In Figure 8-3, the glitch (A) is shorter than the qualification window and is ignored by the input qualifier.



A. 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 is 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT cycles, the GPIO pin will be sampled).

B. The qualification period selected via the GPxCTRL register applies to groups of 8 GPIO pins.

C. The qualification block can take either three or six samples. The QUAL\_SEL Register selects which sample mode is used.

D. 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  $(5 \times \text{QUALPRD} \times 2) \times \text{SYSCLKOUT}$  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.

**Figure 8-3. Input Qualifier Clock Cycles**

## 8.7 GPIO and Peripheral Muxing

### 8.7.1 GPIO 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 works best for your particular application. Refer to [Table 8-6](#) for muxing combinations and definitions.

**Table 8-6. GPIO Muxed Pins**

0, 4, 8, 12	1	2	3	5	6	7	9	10	11	13	14	15	ALT
GPIO0	EPWM1_A	CANA_RX	OUTPUTXBAR7	SCIA_RX	I2CA_SDA	SPIA_STE				EQEP1_INDEX		EPWM3_A	
GPIO1	EPWM1_B			SCIA_TX	I2CA_SCL	SPIA_SOMI	EQEP1_STROBE					EPWM3_B	
GPIO2	EPWM2_A			OUTPUTXBAR1		SPIA_SIMO	SCIA_TX		I2CB_SDA		CANA_TX	EPWM4_A	
GPIO3	EPWM2_B	OUTPUTXBAR2		OUTPUTXBAR2		SPIA_CLK	SCIA_RX		I2CB_SCL		CANA_RX	EPWM4_B	
GPIO4	EPWM3_A	I2CA_SCL		OUTPUTXBAR3	CANA_TX						SPIA_SOMI	EPWM1_A	
GPIO5	EPWM3_B	I2CA_SDA	OUTPUTXBAR3		CANA_RX	SPIA_STE			SCIA_RX			EPWM1_B	
GPIO6	EPWM4_A	OUTPUTXBAR4	SYNCOUT	EQEP1_A								EPWM2_A	
GPIO7	EPWM4_B	EPWM2_A	OUTPUTXBAR5	EQEP1_B		SPIA_SIMO			SCIA_TX		CANA_TX	EPWM2_B	
GPIO8	EPWM5_A		ADCSOAO	EQEP1_STROBE	SCIA_TX	SPIA_SIMO	I2CA_SCL						
GPIO9	EPWM5_B	SCIB_TX	OUTPUTXBAR6	EQEP1_INDEX	SCIA_RX	SPIA_CLK					I2CB_SCL		
GPIO10	EPWM6_A		ADCSOAO	EQEP1_A	SCIB_TX	SPIA_SOMI	I2CA_SDA						
GPIO11	EPWM6_B	CANA_RX	OUTPUTXBAR7	EQEP1_B	SCIB_RX	SPIA_STE				SPIA_SIMO			
GPIO12	EPWM7_A			EQEP1_STROBE	SCIB_TX				SPIA_CLK	CANA_RX			
GPIO13	EPWM7_B			EQEP1_INDEX	SCIB_RX				SPIA_SOMI	CANA_TX			
GPIO16	SPIA_SIMO		OUTPUTXBAR7	EPWM5_A	SCIA_TX		EQEP1_STROBE		XCLKOUT				
GPIO17	SPIA_SOMI		OUTPUTXBAR8	EPWM5_B	SCIA_RX		EQEP1_INDEX		CANA_TX		EPWM6_A		
GPIO18	SPIA_CLK	SCIB_TX	CANA_RX	EPWM6_A	I2CA_SCL				XCLKOUT				X2
GPIO19	SPIA_STE	SCIB_RX	CANA_TX	EPWM6_B	I2CA_SDA								X1
GPIO20	EQEP1_A		CANA_TX		SPIA_SIMO				I2CA_SCL			SCIC_TX	
GPIO21	EQEP1_B		CANA_RX		SPIA_SOMI				I2CA_SDA			SCIC_RX	
GPIO22	EQEP1_STROBE		SCIB_TX				SCIC_TX				EPWM4_A		
GPIO23	EQEP1_INDEX		SCIB_RX				SCIC_RX				EPWM4_B		
GPIO24	OUTPUTXBAR1		SPIA_STE	EPWM4_A	SPIA_SIMO				SCIA_TX	ERRORSTS			
GPIO28	SCIA_RX		EPWM7_A	OUTPUTXBAR5	EQEP1_A			SCIC_TX	SPIA_CLK	ERRORSTS	I2CB_SDA		
GPIO29	SCIA_TX		EPWM7_B	OUTPUTXBAR6	EQEP1_B			SCIC_RX	SPIA_STE	ERRORSTS	I2CB_SCL		
GPIO32	I2CA_SDA	EQEP1_INDEX	SPIA_CLK	EPWM4_B	SCIC_TX				CANA_TX		ADCSOAO		
GPIO33	I2CA_SCL			OUTPUTXBAR4	SCIC_RX				CANA_RX		ADCSOAO		
GPIO35	SCIA_RX	SPIA_SOMI	I2CA_SDA	CANA_RX		SCIC_RX	EQEP1_A		EPWM5_B			TDI	
GPIO37	OUTPUTXBAR2	SPIA_STE	I2CA_SCL	SCIA_TX	CANA_TX	SCIC_TX	EQEP1_B		EPWM5_A			TDO	
GPIO39										SYNCOUT	EQEP1_INDEX		
GPIO40				EPWM2_B			SCIB_TX	EQEP1_A					
GPIO41	EPWM7_A			EPWM2_A			SCIB_RX	EQEP1_B					
GPIO224				OUTPUTXBAR3	SPIA_SIMO		EPWM1_A	CANA_TX	EQEP1_A		SCIC_TX		

**Table 8-6. GPIO Muxed Pins (continued)**

0, 4, 8, 12	1	2	3	5	6	7	9	10	11	13	14	15	ALT
GPIO226				EPWM6_A	SPIA_CLK		EPWM1_B		EQEP1_STROBE		SCIC_RX		
GPIO227	I2CB_SCL		EPWM3_A	OUTPUTXBAR1	EPWM2_B								
GPIO228			ADCSOAO	CANA_TX	SPIA_SOMI		EPWM2_B		EQEP1_B				
GPIO230	I2CB_SDA		EPWM3_B	CANA_RX	EPWM2_A	I2CA_SDA							
GPIO242				OUTPUTXBAR2	SPIA_STE		EPWM4_A	CANA_RX	EQEP1_INDEX				
AIO225													
AIO231													
AIO232													
AIO233													
AIO237													
AIO238													
AIO239													
AIO241													
AIO244													
AIO245													

## 8.7.2 Peripheral Muxing

For example, multiplexing for the GPIO6 pin is controlled by writing to GPAGMUX[13:12] and GPAMUX[13:12]. By writing to these bits, GPIO6 is configured as either a general-purpose digital I/O or one of several different peripheral functions. An example of GPyGMUX and GPyMUX selection and options for a single GPIO are shown in [Table 8-7](#).

### Note

The following table is for example only. Refer to the device data sheet to check the availability of GPIO6 on this device. If GPIO6 is available, the functions mentioned in the table may not match the actual functions available. See [Section 8.7.1](#) for correct list of GPIOs and corresponding mux options for this device.

**Table 8-7. GPIO and Peripheral Muxing**

GPAGMUX1[13:12]	GPAMUX1[13:12]	Pin Functionality
00	00	GPIO6
00	01	Peripheral 1
00	10	Peripheral 2
00	11	Peripheral 3
01	00	GPIO6
01	01	Peripheral 4
01	10	Peripheral 5
01	11	
10	00	GPIO6
10	01	
10	10	Peripheral 6
10	11	Peripheral 7
11	00	GPIO6
11	01	Peripheral 8
11	10	Peripheral 9
11	11	Peripheral 10

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 must not be used.

### CAUTION

If a reserved GPIO mux configuration that is not mapped to either a peripheral or GPIO mode is selected, the state of the pin is undefined and the pin is driven. Unimplemented configurations are for future expansion and must not be selected. In the device mux table (see the data sheet), these options are indicated as Reserved or left blank.

Some peripherals can be assigned to more than one pin by way of the mux registers. For example, OUTPUTXBAR1 can be assigned to GPIOs p, q, or r (where p, q, and r are example GPIO numbers), depending on individual system requirements. An example of this is shown in [Table 8-8](#).

### Note

The following table is for example only. Bit ranges cannot correspond to OUTPUTXBAR1 on this device. See [Section 8.7.1](#) for correct list of GPIOs and corresponding mux options for this device.

If none or more than one of the GPIO pins is configured as peripheral input pins, then that GPIO is set to a hard-wired default value.

**Table 8-8. Peripheral Muxing (Multiple Pins Assigned)**

GMUX Configuration	MUX Configuration	
Choice 1: GPIOp	GPyGMUX1[5:4]=01	GPyMUX1[5:4]=01
or Choice 2: GPIOq	GPyGMUX2[17:16]=00	GPyMUX2[17:16]=01
or Choice 3: GPIOr	GPyGMUX1[7:6]=01	GPyMUX1[7:6]=01

## 8.8 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 must always put each GPIO in one of these configurations:

- Input mode and driven on the board by another component to a level above  $V_{ih}$  or below  $V_{il}$
- Input mode with GPIO internal pullup enabled
- Output mode

On devices with lesser pin count packages, pull-ups on unbonded GPIOs are by default enabled to prevent floating inputs. The user must take care to avoid disabling these pullups in the application code.

On devices with larger pin count packages, the pullups for any internally unbonded GPIO must be enabled to prevent floating inputs. TI has provided functions in controlSUITE/C2000Ware that users can call to enable the pullup on any unbonded GPIO for the package in use. This function, `GPIO_EnabledUnbondedIOPullups()`, resides in the `(Device)_Sysctrl.c` file and is called by default from `InitSysCtrl()`. The user must take care to avoid disabling these pullups in the application code.

## 8.9 Software

### 8.9.1 GPIO Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/gpio

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](https://dev.ti.com/C2000Ware/Examples).

#### 8.9.1.1 Device GPIO Setup

FILE: gpio\_ex1\_setup.c

Configures the device GPIO into two different configurations This code is verbose to illustrate how the GPIO could be setup. In a real application, lines of code can be combined for improved code size and efficiency.

This example only sets-up the GPIO. Nothing is actually done with the pins after setup.

*In general:*

- All pullup resistors are enabled. For ePWMs this may not be desired.
- Input qual for communication ports (SPI, SCI, I2C) is asynchronous
- Input qual for Trip pins (TZ) is asynchronous
- Input qual for eCAP and eQEP signals is synch to SYSCLKOUT
- Input qual for some I/O's and \_\_interrupts may have a sampling window

#### 8.9.1.2 Device GPIO Toggle

FILE: gpio\_ex2\_toggle.c

Configures the device GPIO through the sysconfig file. The GPIO pin is toggled in the infinite loop. In order to migrate the project within syscfg to any device, click the switch button under the device view and select your corresponding device to migrate, saving the project will auto-migrate your project settings.

#### 8.9.1.3 Device GPIO Interrupt

FILE: gpio\_ex3\_interrupt.c

Configures the device GPIOs through the sysconfig file. One GPIO output pin, and one GPIO input pin is configured. The example then configures the GPIO input pin to be the source of an external interrupt which toggles the GPIO output pin.

#### 8.9.1.4 External Interrupt (XINT)

FILE: gpio\_ex4\_aio\_external\_interrupt.c

In this example AIO pins are configured as digital inputs. Two other GPIO signals (connected externally to AIO pins) are toggled in software to trigger external interrupt through AIO224 and AIO225 (AIO224 assigned to XINT1 and AIO225 assigned to XINT2). The user is required to externally connect these signals for the program to work properly. Each interrupt is fired in sequence: XINT1 first and then XINT2.

- GPIO5 will go high outside of the interrupts and low within the interrupts. This signal can be monitored on a scope. *ExternalConnections*
- Connect GPIO0 to AIO224. AIO224 will be assigned to XINT1
- Connect GPIO1 to AIO225. AIO225 will be assigned to XINT2
- GPIO5 can be monitored on an oscilloscope

*Watch Variables*

- xint1Count for the number of times through XINT1 interrupt
- xint2Count for the number of times through XINT2 interrupt
- loopCount for the number of times through the idle loop

## 8.9.2 LED Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
 C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/led

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

## 8.10 GPIO Registers

This section describes the General-Purpose Input/Output Registers.

### 8.10.1 GPIO Base Address Table

**Table 8-9. GPIO Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
GpioCtrlRegs	GPIO_CTRL_REGS	GPIOCTRL_BASE	0x0000_7C00	YES
GpioDataRegs	GPIO_DATA_REGS	GPIODATA_BASE	0x0000_7F00	YES
GpioDataReadRegs	GPIO_DATA_READ_REGS	GPIODATAREAD_BASE	0x0000_7F80	YES

## 8.10.2 GPIO\_CTRL\_REGS Registers

Table 8-10 lists the memory-mapped registers for the GPIO\_CTRL\_REGS registers. All register offset addresses not listed in Table 8-10 should be considered as reserved locations and the register contents should not be modified.

**Table 8-10. GPIO\_CTRL\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GPACTRL	GPIO A Qualification Sampling Period Control (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
2h	GPAQSEL1	GPIO A Qualifier Select 1 Register (GPIO0 to 15)	EALLOW	<a href="#">Go</a>
4h	GPAQSEL2	GPIO A Qualifier Select 2 Register (GPIO16 to 31)	EALLOW	<a href="#">Go</a>
6h	GPAMUX1	GPIO A Mux 1 Register (GPIO0 to 15)	EALLOW	<a href="#">Go</a>
8h	GPAMUX2	GPIO A Mux 2 Register (GPIO16 to 31)	EALLOW	<a href="#">Go</a>
Ah	GPADIR	GPIO A Direction Register (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
Ch	GPAPUD	GPIO A Pull Up Disable Register (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
10h	GPAINV	GPIO A Input Polarity Invert Registers (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
12h	GPAODR	GPIO A Open Drain Output Register (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
14h	GPAAMSEL	GPIO A Analog Mode Select register (GPIO0 to GPIO31)	EALLOW	<a href="#">Go</a>
20h	GPAGMUX1	GPIO A Peripheral Group Mux (GPIO0 to 15)	EALLOW	<a href="#">Go</a>
22h	GPAGMUX2	GPIO A Peripheral Group Mux (GPIO16 to 31)	EALLOW	<a href="#">Go</a>
3Ch	GPALOCK	GPIO A Lock Configuration Register (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
3Eh	GPACR	GPIO A Lock Commit Register (GPIO0 to 31)	EALLOW	<a href="#">Go</a>
40h	GPBCTRL	GPIO B Qualification Sampling Period Control (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
42h	GPBQSEL1	GPIO B Qualifier Select 1 Register (GPIO32 to 47)	EALLOW	<a href="#">Go</a>
46h	GPBMUX1	GPIO B Mux 1 Register (GPIO32 to 47)	EALLOW	<a href="#">Go</a>
4Ah	GPBDIR	GPIO B Direction Register (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
4Ch	GPBPUD	GPIO B Pull Up Disable Register (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
50h	GPBINV	GPIO B Input Polarity Invert Registers (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
52h	GPBODR	GPIO B Open Drain Output Register (GPIO32 to GPIO63)	EALLOW	<a href="#">Go</a>
60h	GPBGMUX1	GPIO B Peripheral Group Mux (GPIO32 to 47)	EALLOW	<a href="#">Go</a>
7Ch	GPBLOCK	GPIO B Lock Configuration Register (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
7Eh	GPBCR	GPIO B Lock Commit Register (GPIO32 to 63)	EALLOW	<a href="#">Go</a>
1C0h	GPHCTRL	GPIO H Qualification Sampling Period Control (GPIO224 to 255)	EALLOW	<a href="#">Go</a>
1C2h	GPHQSEL1	GPIO H Qualifier Select 1 Register (GPIO224 to 239)	EALLOW	<a href="#">Go</a>
1C4h	GPHQSEL2	GPIO H Qualifier Select 2 Register (GPIO240 to 255)	EALLOW	<a href="#">Go</a>
1C6h	GPHMUX1	GPIO H Mux 1 Register (GPIO224 to 239)	EALLOW	<a href="#">Go</a>
1C8h	GPHMUX2	GPIO H Mux 2 Register (GPIO240 to 255)	EALLOW	<a href="#">Go</a>
1CAh	GPHDIR	GPIO H Direction Register (GPIO224 to 255)	EALLOW	<a href="#">Go</a>

**Table 8-10. GPIO\_CTRL\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
1CCh	GPHPUD	GPIO H Pull Up Disable Register (GPIO224 to 255)	EALLOW	<a href="#">Go</a>
1D0h	GPHINV	GPIO H Input Polarity Invert Registers (GPIO224 to 255)	EALLOW	<a href="#">Go</a>
1D2h	GPHODR	GPIO H Open Drain Output Register (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1D4h	GPHAMSEL	GPIO H Analog Mode Select register (GPIO224 to GPIO255)	EALLOW	<a href="#">Go</a>
1E0h	GPHGMUX1	GPIO H Peripheral Group Mux (GPIO224 to 239)	EALLOW	<a href="#">Go</a>
1E2h	GPHGMUX2	GPIO H Peripheral Group Mux (GPIO240 to 255)	EALLOW	<a href="#">Go</a>
1FCh	GPHLOCK	GPIO H Lock Configuration Register (GPIO224 to 255)	EALLOW	<a href="#">Go</a>
1FEh	GPHCR	GPIO H Lock Commit Register (GPIO224 to 255)	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 8-11](#) shows the codes that are used for access types in this section.

**Table 8-11. GPIO\_CTRL\_REGS Access Type Codes**

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

### 8.10.2.1 GACTRL Register (Offset = 0h) [Reset = 0000000h]

GACTRL is shown in [Figure 8-4](#) and described in [Table 8-12](#).

Return to the [Summary Table](#).

GPIO A Qualification Sampling Period Control (GPIO0 to 31)

**Figure 8-4. GACTRL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QUALPRD3								QUALPRD2								QUALPRD1								QUALPRD0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 8-12. GACTRL 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 Reset type: SYSRSn
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 Reset type: SYSRSn
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 Reset type: SYSRSn
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 Reset type: SYSRSn

### 8.10.2.2 GPAQSEL1 Register (Offset = 2h) [Reset = 0000000h]

GPAQSEL1 is shown in [Figure 8-5](#) and described in [Table 8-13](#).

Return to the [Summary Table](#).

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 8-5. GPAQSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-13. GPAQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO13	R/W	0h	Select input qualification type for GPIO13: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Select input qualification type for GPIO12: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Select input qualification type for GPIO11: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Select input qualification type for GPIO10: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Select input qualification type for GPIO9: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

**Table 8-13. GPAQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17-16	GPIO8	R/W	0h	Select input qualification type for GPIO8: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Select input qualification type for GPIO7: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Select input qualification type for GPIO6: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Select input qualification type for GPIO5: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Select input qualification type for GPIO4: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Select input qualification type for GPIO3: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Select input qualification type for GPIO2: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
3-2	GPIO1	R/W	0h	Select input qualification type for GPIO1: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
1-0	GPIO0	R/W	0h	Select input qualification type for GPIO0: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

### 8.10.2.3 GPAQSEL2 Register (Offset = 4h) [Reset = 0000000h]

GPAQSEL2 is shown in [Figure 8-6](#) and described in [Table 8-14](#).

Return to the [Summary Table](#).

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 8-6. GPAQSEL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO29		GPIO28		RESERVED		RESERVED		RESERVED		GPIO24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-14. GPAQSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO29	R/W	0h	Select input qualification type for GPIO29: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Select input qualification type for GPIO28: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	GPIO24	R/W	0h	Select input qualification type for GPIO24: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Select input qualification type for GPIO23: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

**Table 8-14. GPAQSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-12	GPIO22	R/W	0h	Select input qualification type for GPIO22: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Select input qualification type for GPIO21: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Select input qualification type for GPIO20: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Select input qualification type for GPIO19: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Select input qualification type for GPIO18: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
3-2	GPIO17	R/W	0h	Select input qualification type for GPIO17: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
1-0	GPIO16	R/W	0h	Select input qualification type for GPIO16: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

### 8.10.2.4 GPAMUX1 Register (Offset = 6h) [Reset = 0000000h]

GPAMUX1 is shown in [Figure 8-7](#) and described in [Table 8-15](#).

Return to the [Summary Table](#).

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. Refer to GPIO chapter for more details.

**Figure 8-7. GPAMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-15. GPAMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO13	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
17-16	GPIO8	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	GPIO1	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO0	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.5 GPAMUX2 Register (Offset = 8h) [Reset = 0000000h]

GPAMUX2 is shown in [Figure 8-8](#) and described in [Table 8-16](#).

Return to the [Summary Table](#).

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. Refer to GPIO chapter for more details.

**Figure 8-8. GPAMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO29		GPIO28		RESERVED		RESERVED		RESERVED		GPIO24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-16. GPAMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO29	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	GPIO24	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	GPIO22	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	GPIO17	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO16	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.6 GPADIR Register (Offset = Ah) [Reset = 0000000h]

GPADIR is shown in [Figure 8-9](#) and described in [Table 8-17](#).

Return to the [Summary Table](#).

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 8-9. GPADIR Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-17. GPADIR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	GPIO29	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
28	GPIO28	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	GPIO24	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
23	GPIO23	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
22	GPIO22	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
21	GPIO21	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
20	GPIO20	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
19	GPIO19	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
18	GPIO18	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn

**Table 8-17. GPADIR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
16	GPIO16	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
12	GPIO12	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
11	GPIO11	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
10	GPIO10	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
9	GPIO9	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
8	GPIO8	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
7	GPIO7	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
6	GPIO6	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
5	GPIO5	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
4	GPIO4	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
3	GPIO3	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
2	GPIO2	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
1	GPIO1	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
0	GPIO0	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn

### 8.10.2.7 GPAPUD Register (Offset = Ch) [Reset = FFFFFFFFh]

GPAPUD is shown in [Figure 8-10](#) and described in [Table 8-18](#).

Return to the [Summary Table](#).

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.

Note:

[1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low.

**Figure 8-10. GPAPUD Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

**Table 8-18. GPAPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	GPIO29	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
28	GPIO28	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	GPIO24	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
23	GPIO23	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
22	GPIO22	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
21	GPIO21	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
20	GPIO20	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
19	GPIO19	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn

**Table 8-18. GPAPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO18	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
17	GPIO17	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
16	GPIO16	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
15	RESERVED	R/W	1h	Reserved
14	RESERVED	R/W	1h	Reserved
13	GPIO13	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
12	GPIO12	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
11	GPIO11	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
10	GPIO10	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
9	GPIO9	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
8	GPIO8	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
7	GPIO7	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
6	GPIO6	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
5	GPIO5	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
4	GPIO4	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
3	GPIO3	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
2	GPIO2	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
1	GPIO1	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
0	GPIO0	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn

### 8.10.2.8 GPAINV Register (Offset = 10h) [Reset = 0000000h]

GPAINV is shown in [Figure 8-11](#) and described in [Table 8-19](#).

Return to the [Summary Table](#).

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 8-11. GPAINV Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-19. GPAINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	GPIO29	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
28	GPIO28	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	GPIO24	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
23	GPIO23	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
22	GPIO22	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
21	GPIO21	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
20	GPIO20	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
19	GPIO19	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
18	GPIO18	R/W	0h	Input inversion control for this pin Reset type: SYSRSn

**Table 8-19. GPAINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
16	GPIO16	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
12	GPIO12	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
11	GPIO11	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
10	GPIO10	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
9	GPIO9	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
8	GPIO8	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
7	GPIO7	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
6	GPIO6	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
5	GPIO5	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
4	GPIO4	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
3	GPIO3	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
2	GPIO2	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
1	GPIO1	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
0	GPIO0	R/W	0h	Input inversion control for this pin Reset type: SYSRSn

### 8.10.2.9 GPAODR Register (Offset = 12h) [Reset = 0000000h]

GPAODR is shown in [Figure 8-12](#) and described in [Table 8-20](#).

Return to the [Summary Table](#).

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 8-12. GPAODR Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-20. GPAODR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	GPIO29	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
28	GPIO28	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	GPIO24	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
23	GPIO23	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
22	GPIO22	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
21	GPIO21	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
20	GPIO20	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn

**Table 8-20. GPAODR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
18	GPIO18	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
17	GPIO17	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
16	GPIO16	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
12	GPIO12	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
11	GPIO11	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
10	GPIO10	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
9	GPIO9	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
8	GPIO8	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
7	GPIO7	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
6	GPIO6	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
5	GPIO5	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
4	GPIO4	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
3	GPIO3	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
2	GPIO2	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
1	GPIO1	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
0	GPIO0	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn

### 8.10.2.10 GPAAMSEL Register (Offset = 14h) [Reset = FFFFFFFFh]

GPAAMSEL is shown in [Figure 8-13](#) and described in [Table 8-21](#).

Return to the [Summary Table](#).

GPIO A Analog Mode Select register (GPIO0 to GPIO31)

Selects between digital and analog functionality for GPIO pins.

0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers

1: The analog function of the pin is enabled and the pin is capable of analog functions

Reading the register returns the current value of the register setting.

Note:

[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers don't have any affect.

**Figure 8-13. GPAAMSEL Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	GPIO28	RESERVED	RESERVED	RESERVED	RESERVED
R/W-1h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO21	GPIO20	RESERVED	RESERVED	RESERVED	RESERVED
R/W-1h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	RESERVED	RESERVED	RESERVED	RESERVED
R/W-1h							
7	6	5	4	3	2	1	0
RESERVED							
R/W-1h							

**Table 8-21. GPAAMSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	GPIO28	R/W	1h	Analog Mode select for this pin Reset type: SYSRSn
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	RESERVED	R/W	1h	Reserved
22	RESERVED	R/W	1h	Reserved
21	GPIO21	R/W	1h	Analog Mode select for this pin Reset type: SYSRSn
20	GPIO20	R/W	1h	Analog Mode select for this pin Reset type: SYSRSn
19	RESERVED	R/W	1h	Reserved
18	RESERVED	R/W	1h	Reserved
17	RESERVED	R/W	1h	Reserved
16	RESERVED	R/W	1h	Reserved

**Table 8-21. GPAAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	RESERVED	R/W	1h	Reserved
14	RESERVED	R/W	1h	Reserved
13	GPIO13	R/W	1h	Analog Mode select for this pin Reset type: SYSRSn
12	GPIO12	R/W	1h	Analog Mode select for this pin Reset type: SYSRSn
11	RESERVED	R/W	1h	Reserved
10	RESERVED	R/W	1h	Reserved
9	RESERVED	R/W	1h	Reserved
8	RESERVED	R/W	1h	Reserved
7	RESERVED	R/W	1h	Reserved
6	RESERVED	R/W	1h	Reserved
5	RESERVED	R/W	1h	Reserved
4	RESERVED	R/W	1h	Reserved
3	RESERVED	R/W	1h	Reserved
2	RESERVED	R/W	1h	Reserved
1	RESERVED	R/W	1h	Reserved
0	RESERVED	R/W	1h	Reserved

### 8.10.2.11 GPAGMUX1 Register (Offset = 20h) [Reset = 0000000h]

GPAGMUX1 is shown in [Figure 8-14](#) and described in [Table 8-22](#).

Return to the [Summary Table](#).

GPIO A Peripheral Group Mux (GPIO0 to 15)

Defines pin-muxing selection for GPIO.

Notes:

[1]For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-14. GPAGMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO13		GPIO12		GPIO11		GPIO10		GPIO9		GPIO8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO7		GPIO6		GPIO5		GPIO4		GPIO3		GPIO2		GPIO1		GPIO0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-22. GPAGMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO13	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
25-24	GPIO12	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
23-22	GPIO11	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
21-20	GPIO10	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
19-18	GPIO9	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
17-16	GPIO8	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO7	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	GPIO6	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	GPIO5	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	GPIO4	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO3	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO2	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	GPIO1	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO0	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.12 GPAGMUX2 Register (Offset = 22h) [Reset = 0000000h]

GPAGMUX2 is shown in [Figure 8-15](#) and described in [Table 8-23](#).

Return to the [Summary Table](#).

GPIO A Peripheral Group Mux (GPIO16 to 31)

Defines pin-muxing selection for GPIO.

Notes:

[1]For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-15. GPAGMUX2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO29		GPIO28		RESERVED		RESERVED		RESERVED		GPIO24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO23		GPIO22		GPIO21		GPIO20		GPIO19		GPIO18		GPIO17		GPIO16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-23. GPAGMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	GPIO29	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
25-24	GPIO28	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	GPIO24	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO23	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	GPIO22	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	GPIO21	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	GPIO20	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO19	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO18	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	GPIO17	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO16	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.13 GPALOCK Register (Offset = 3Ch) [Reset = 0000000h]

GPALOCK is shown in [Figure 8-16](#) and described in [Table 8-24](#).

Return to the [Summary Table](#).

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 8-16. GPALOCK Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-24. GPALOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	GPIO29	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
28	GPIO28	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	GPIO24	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
23	GPIO23	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
22	GPIO22	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
21	GPIO21	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
20	GPIO20	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
19	GPIO19	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn

**Table 8-24. GPALOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO18	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
17	GPIO17	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
16	GPIO16	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
12	GPIO12	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
11	GPIO11	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
10	GPIO10	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
9	GPIO9	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
8	GPIO8	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
7	GPIO7	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
6	GPIO6	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
5	GPIO5	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
4	GPIO4	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
3	GPIO3	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
2	GPIO2	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
1	GPIO1	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
0	GPIO0	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn

### 8.10.2.14 GPACR Register (Offset = 3Eh) [Reset = 0000000h]

GPACR is shown in [Figure 8-17](#) and described in [Table 8-25](#).

Return to the [Summary Table](#).

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 8-17. GPACR Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	GPIO24
R/WOnce-0h							
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/WOnce-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/WOnce-0h							
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/WOnce-0h							

**Table 8-25. GPACR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	GPIO29	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
28	GPIO28	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
27	RESERVED	R/WOnce	0h	Reserved
26	RESERVED	R/WOnce	0h	Reserved
25	RESERVED	R/WOnce	0h	Reserved
24	GPIO24	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
23	GPIO23	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
22	GPIO22	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
21	GPIO21	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
20	GPIO20	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
19	GPIO19	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
18	GPIO18	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn

**Table 8-25. GPACR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
16	GPIO16	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
15	RESERVED	R/WOnce	0h	Reserved
14	RESERVED	R/WOnce	0h	Reserved
13	GPIO13	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
12	GPIO12	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
11	GPIO11	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
10	GPIO10	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
9	GPIO9	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
8	GPIO8	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
7	GPIO7	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
6	GPIO6	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
5	GPIO5	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
4	GPIO4	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
3	GPIO3	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
2	GPIO2	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
1	GPIO1	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
0	GPIO0	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn

### 8.10.2.15 GPBCTRL Register (Offset = 40h) [Reset = 0000000h]

GPBCTRL is shown in [Figure 8-18](#) and described in [Table 8-26](#).

Return to the [Summary Table](#).

GPIO B Qualification Sampling Period Control (GPIO32 to 63)

**Figure 8-18. 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
RESERVED								RESERVED								QUALPRD1				QUALPRD0											
R/W-0h								R/W-0h								R/W-0h				R/W-0h											

**Table 8-26. GPBCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	0h	Reserved
23-16	RESERVED	R/W	0h	Reserved
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 Reset type: SYSRSn
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 Reset type: SYSRSn

### 8.10.2.16 GPBQSEL1 Register (Offset = 42h) [Reset = 0000CC0h]

GPBQSEL1 is shown in [Figure 8-19](#) and described in [Table 8-27](#).

Return to the [Summary Table](#).

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 8-19. GPBQSEL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		GPIO41		GPIO40	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		RESERVED		GPIO33		GPIO32	
R/W-0h		R/W-0h		R/W-3h		R/W-0h		R/W-3h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-27. GPBQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	GPIO41	R/W	0h	Select input qualification type for GPIO41: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Select input qualification type for GPIO40: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Select input qualification type for GPIO39: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	3h	Select input qualification type for GPIO37: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved

**Table 8-27. GPBQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	GPIO35	R/W	3h	Select input qualification type for GPIO35: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
5-4	RESERVED	R/W	0h	Reserved
3-2	GPIO33	R/W	0h	Select input qualification type for GPIO33: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
1-0	GPIO32	R/W	0h	Select input qualification type for GPIO32: 0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

### 8.10.2.17 GPBMUX1 Register (Offset = 46h) [Reset = 0000CC0h]

GPBMUX1 is shown in [Figure 8-20](#) and described in [Table 8-28](#).

Return to the [Summary Table](#).

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. Refer to GPIO chapter for more details.

**Figure 8-20. GPBMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		GPIO41		GPIO40	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		RESERVED		GPIO33		GPIO32	
R/W-0h		R/W-0h		R/W-3h		R/W-0h		R/W-3h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-28. GPBMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	GPIO41	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	3h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved
7-6	GPIO35	R/W	3h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	RESERVED	R/W	0h	Reserved
3-2	GPIO33	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO32	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.18 GPBDIR Register (Offset = 4Ah) [Reset = 0000000h]

GPBDIR is shown in [Figure 8-21](#) and described in [Table 8-29](#).

Return to the [Summary Table](#).

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 8-21. GPBDIR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-0h							

**Table 8-29. GPBDIR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved

**Table 8-29. GPBDIR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
8	GPIO40	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
7	GPIO39	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	GPIO33	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
0	GPIO32	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn

### 8.10.2.19 GPBPUD Register (Offset = 4Ch) [Reset = FFFFFFFFh]

GPBPUD is shown in [Figure 8-22](#) and described in [Table 8-30](#).

Return to the [Summary Table](#).

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.

Note:

[1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low.

**Figure 8-22. GPBPUD Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-1h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-1h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-1h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-1h							

**Table 8-30. GPBPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	RESERVED	R/W	1h	Reserved
22	RESERVED	R/W	1h	Reserved
21	RESERVED	R/W	1h	Reserved
20	RESERVED	R/W	1h	Reserved
19	RESERVED	R/W	1h	Reserved
18	RESERVED	R/W	1h	Reserved
17	RESERVED	R/W	1h	Reserved
16	RESERVED	R/W	1h	Reserved
15	RESERVED	R/W	1h	Reserved
14	RESERVED	R/W	1h	Reserved
13	RESERVED	R/W	1h	Reserved
12	RESERVED	R/W	1h	Reserved

**Table 8-30. GPBPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	RESERVED	R/W	1h	Reserved
10	RESERVED	R/W	1h	Reserved
9	GPIO41	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
8	GPIO40	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
7	GPIO39	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
6	RESERVED	R/W	1h	Reserved
5	GPIO37	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
4	RESERVED	R/W	1h	Reserved
3	GPIO35	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
2	RESERVED	R/W	1h	Reserved
1	GPIO33	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn
0	GPIO32	R/W	1h	Pull-Up Disable control for this pin Reset type: SYSRSn

### 8.10.2.20 GPBINV Register (Offset = 50h) [Reset = 0000000h]

GPBINV is shown in [Figure 8-23](#) and described in [Table 8-31](#).

Return to the [Summary Table](#).

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 8-23. GPBINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-0h							

**Table 8-31. GPBINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved

**Table 8-31. GPBINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
8	GPIO40	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
7	GPIO39	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	GPIO33	R/W	0h	Input inversion control for this pin Reset type: SYSRSn
0	GPIO32	R/W	0h	Input inversion control for this pin Reset type: SYSRSn

### 8.10.2.21 GPBODR Register (Offset = 52h) [Reset = 0000000h]

GPBODR is shown in [Figure 8-24](#) and described in [Table 8-32](#).

Return to the [Summary Table](#).

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 8-24. GPBODR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-0h							

**Table 8-32. GPBODR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved

**Table 8-32. GPBODR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	GPIO41	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
8	GPIO40	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
7	GPIO39	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	GPIO33	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
0	GPIO32	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn

### 8.10.2.2 GPBGMUX1 Register (Offset = 60h) [Reset = 0000CC0h]

GPBGMUX1 is shown in [Figure 8-25](#) and described in [Table 8-33](#).

Return to the [Summary Table](#).

GPIO B Peripheral Group Mux (GPIO32 to 47)

Defines pin-muxing selection for GPIO.

Notes:

[1]For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-25. GPBGMUX1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		RESERVED		GPIO41		GPIO40	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO39		RESERVED		GPIO37		RESERVED		GPIO35		RESERVED		GPIO33		GPIO32	
R/W-0h		R/W-0h		R/W-3h		R/W-0h		R/W-3h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-33. GPBGMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	GPIO41	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
17-16	GPIO40	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
15-14	GPIO39	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO37	R/W	3h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
9-8	RESERVED	R/W	0h	Reserved
7-6	GPIO35	R/W	3h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	RESERVED	R/W	0h	Reserved
3-2	GPIO33	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
1-0	GPIO32	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.23 GPBLOCK Register (Offset = 7Ch) [Reset = 0000000h]

GPBLOCK is shown in [Figure 8-26](#) and described in [Table 8-34](#).

Return to the [Summary Table](#).

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 8-26. GPBLOCK Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-0h							

**Table 8-34. GPBLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved

**Table 8-34. GPBLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	RESERVED	R/W	0h	Reserved
9	GPIO41	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
8	GPIO40	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
7	GPIO39	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	GPIO33	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn
0	GPIO32	R/W	0h	Configuration Lock bit for this pin Reset type: SYSRSn

### 8.10.2.24 GPBCR Register (Offset = 7Eh) [Reset = 0000000h]

GPBCR is shown in [Figure 8-27](#) and described in [Table 8-35](#).

Return to the [Summary Table](#).

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 8-27. GPBCR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/WOnce-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/WOnce-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/WOnce-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/WOnce-0h							

**Table 8-35. GPBCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	RESERVED	R/WOnce	0h	Reserved
28	RESERVED	R/WOnce	0h	Reserved
27	RESERVED	R/WOnce	0h	Reserved
26	RESERVED	R/WOnce	0h	Reserved
25	RESERVED	R/WOnce	0h	Reserved
24	RESERVED	R/WOnce	0h	Reserved
23	RESERVED	R/WOnce	0h	Reserved
22	RESERVED	R/WOnce	0h	Reserved
21	RESERVED	R/WOnce	0h	Reserved
20	RESERVED	R/WOnce	0h	Reserved
19	RESERVED	R/WOnce	0h	Reserved
18	RESERVED	R/WOnce	0h	Reserved
17	RESERVED	R/WOnce	0h	Reserved
16	RESERVED	R/WOnce	0h	Reserved
15	RESERVED	R/WOnce	0h	Reserved
14	RESERVED	R/WOnce	0h	Reserved
13	RESERVED	R/WOnce	0h	Reserved
12	RESERVED	R/WOnce	0h	Reserved
11	RESERVED	R/WOnce	0h	Reserved
10	RESERVED	R/WOnce	0h	Reserved

**Table 8-35. GPBCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
8	GPIO40	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
7	GPIO39	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
6	RESERVED	R/WOnce	0h	Reserved
5	GPIO37	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
4	RESERVED	R/WOnce	0h	Reserved
3	GPIO35	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
2	RESERVED	R/WOnce	0h	Reserved
1	GPIO33	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn
0	GPIO32	R/WOnce	0h	Configuration lock commit bit for this pin Reset type: SYSRSn

### 8.10.2.25 GPHCTRL Register (Offset = 1C0h) [Reset = 0000000h]

GPHCTRL is shown in [Figure 8-28](#) and described in [Table 8-36](#).

Return to the [Summary Table](#).

GPIO H Qualification Sampling Period Control (GPIO224 to 255)

**Figure 8-28. GPHCTRL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								QUALPRD2								QUALPRD1								QUALPRD0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 8-36. GPHCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	0h	Reserved
23-16	QUALPRD2	R/W	0h	Qualification sampling period for GPIO240 to GPIO247: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 .... 0xFF,QUALPRDx = PLLSYSCLK/512 Reset type: SYSRSn
15-8	QUALPRD1	R/W	0h	Qualification sampling period for GPIO232 to GPIO239: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 .... 0xFF,QUALPRDx = PLLSYSCLK/511 Reset type: SYSRSn
7-0	QUALPRD0	R/W	0h	Qualification sampling period for GPIO224 to GPIO231: 0x00,QUALPRDx = PLLSYSCLK 0x01,QUALPRDx = PLLSYSCLK/2 0x02,QUALPRDx = PLLSYSCLK/4 .... 0xFF,QUALPRDx = PLLSYSCLK/510 Reset type: SYSRSn

### 8.10.2.26 GPHQSEL1 Register (Offset = 1C2h) [Reset = 0000000h]

GPHQSEL1 is shown in [Figure 8-29](#) and described in [Table 8-37](#).

Return to the [Summary Table](#).

GPIO H Qualifier Select 1 Register (GPIO224 to 239)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

**Figure 8-29. GPHQSEL1 Register**

31	30	29	28	27	26	25	24
GPIO239		GPIO238		GPIO237		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		GPIO233		GPIO232	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
GPIO231		GPIO230		RESERVED		GPIO228	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO227		GPIO226		GPIO225		GPIO224	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-37. GPHQSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	GPIO239	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
29-28	GPIO238	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
27-26	GPIO237	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	GPIO233	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

**Table 8-37. GPHQSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17-16	GPIO232	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
15-14	GPIO231	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
13-12	GPIO230	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
11-10	RESERVED	R/W	0h	Reserved
9-8	GPIO228	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
7-6	GPIO227	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
5-4	GPIO226	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
3-2	GPIO225	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
1-0	GPIO224	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn

### 8.10.2.27 GPHQSEL2 Register (Offset = 1C4h) [Reset = 0000000h]

GPHQSEL2 is shown in [Figure 8-30](#) and described in [Table 8-38](#).

Return to the [Summary Table](#).

GPIO H Qualifier Select 2 Register (GPIO240 to 255)

Input qualification type:

0,0 Sync

0,1 Qualification (3 samples)

1,0 Qualification (6 samples)

1,1 Async (no Sync or Qualification)

**Figure 8-30. GPHQSEL2 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
RESERVED		RESERVED		GPIO245		GPIO244	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		GPIO242		GPIO241		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-38. GPHQSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	GPIO245	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
9-8	GPIO244	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
7-6	RESERVED	R/W	0h	Reserved

**Table 8-38. GPHQSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-4	GPIO242	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
3-2	GPIO241	R/W	0h	0,0,Sync 0,1,Qualification (3 samples) 1,0,Qualification (6 samples) 1,1,Async (no Sync or Qualification) Reset type: SYSRSn
1-0	RESERVED	R/W	0h	Reserved

### 8.10.2.28 GPHMUX1 Register (Offset = 1C6h) [Reset = 0000000h]

GPHMUX1 is shown in [Figure 8-31](#) and described in [Table 8-39](#).

Return to the [Summary Table](#).

GPIO H Mux 1 Register (GPIO224 to 239)

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. Refer to GPIO chapter for more details.

**Figure 8-31. GPHMUX1 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
RESERVED		GPIO230		RESERVED		GPIO228	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO227		GPIO226		RESERVED		GPIO224	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-39. GPHMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	GPIO230	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	RESERVED	R/W	0h	Reserved
9-8	GPIO228	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO227	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO226	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	RESERVED	R/W	0h	Reserved
1-0	GPIO224	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.29 GPHMUX2 Register (Offset = 1C8h) [Reset = 0000000h]

GPHMUX2 is shown in [Figure 8-32](#) and described in [Table 8-40](#).

Return to the [Summary Table](#).

GPIO H Mux 2 Register (GPIO240 to 255)

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. Refer to GPIO chapter for more details.

**Figure 8-32. GPHMUX2 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		GPIO242		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-40. GPHMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R/W	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	GPIO242	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	RESERVED	R/W	0h	Reserved
1-0	RESERVED	R/W	0h	Reserved

### 8.10.2.30 GPHDIR Register (Offset = 1CAh) [Reset = 0000000h]

GPHDIR is shown in [Figure 8-33](#) and described in [Table 8-41](#).

Return to the [Summary Table](#).

GPIO H Direction Register (GPIO224 to 255)

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 8-33. GPHDIR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R/W-0h							

**Table 8-41. GPHDIR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	GPIO242	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved

**Table 8-41. GPHDIR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	GPIO230	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
3	GPIO227	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
2	GPIO226	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	GPIO224	R/W	0h	Defines direction for this pin in GPIO mode Reset type: SYSRSn

### 8.10.2.31 GPHPUD Register (Offset = 1CCh) [Reset = FFFFFFFFh]

GPHPUD is shown in [Figure 8-34](#) and described in [Table 8-42](#).

Return to the [Summary Table](#).

GPIO H Pull Up Disable Register (GPIO224 to 255)

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.

Note:

[1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low.

**Figure 8-34. GPHPUD Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-1h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/W-1h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/W-1h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/W-1h							

**Table 8-42. GPHPUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	RESERVED	R/W	1h	Reserved
22	RESERVED	R/W	1h	Reserved
21	GPIO245	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn

**Table 8-42. GPHPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	GPIO244	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
19	RESERVED	R/W	1h	Reserved
18	GPIO242	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
17	GPIO241	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
16	RESERVED	R/W	1h	Reserved
15	GPIO239	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
14	GPIO238	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
13	GPIO237	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
12	RESERVED	R/W	1h	Reserved
11	RESERVED	R/W	1h	Reserved
10	RESERVED	R/W	1h	Reserved

**Table 8-42. GPHPU Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO233	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
8	GPIO232	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
7	GPIO231	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
6	GPIO230	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
5	RESERVED	R/W	1h	Reserved
4	GPIO228	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
3	GPIO227	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn

**Table 8-42. GPHPUD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	GPIO226	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
1	GPIO225	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn
0	GPIO224	R/W	1h	0: Enables the Pull-Up. 1: Disables the Pull-Up. Reading the register returns the current value of the register setting. Note: [1] The Pull-Ups on the GPIO pins are disabled asynchronously when IORSn signal is low. When coming out of reset, the pull-ups will remain disabled until the user enables them selectively in software by writing to this register. Reset type: SYSRSn

### 8.10.2.32 GPHINV Register (Offset = 1D0h) [Reset = 0000000h]

GPHINV is shown in [Figure 8-35](#) and described in [Table 8-43](#).

Return to the [Summary Table](#).

GPIO H Input Polarity Invert Registers (GPIO224 to 255)

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 8-35. GPHINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/W-0h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/W-0h							

**Table 8-43. GPHINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	GPIO245	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
20	GPIO244	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
19	RESERVED	R/W	0h	Reserved

**Table 8-43. GPHINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	GPIO242	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
17	GPIO241	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
16	RESERVED	R/W	0h	Reserved
15	GPIO239	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
14	GPIO238	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
13	GPIO237	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	GPIO233	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
8	GPIO232	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
7	GPIO231	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn

**Table 8-43. GPHINV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	GPIO230	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
3	GPIO227	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
2	GPIO226	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
1	GPIO225	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn
0	GPIO224	R/W	0h	0: selects non-inverted GPIO input 1: selects inverted GPIO input Notes: [1] Reading the register returns the current value of the register setting. Reset type: SYSRSn

### 8.10.2.33 GPHODR Register (Offset = 1D2h) [Reset = 0000000h]

GPHODR is shown in [Figure 8-36](#) and described in [Table 8-44](#).

Return to the [Summary Table](#).

GPIO H Open Drain Output Register (GPIO224 to GPIO255)

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 8-36. GPHODR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R/W-0h							

**Table 8-44. GPHODR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	GPIO242	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved

**Table 8-44. GPHODR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	GPIO230	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
3	GPIO227	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
2	GPIO226	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn
1	RESERVED	R/W	0h	Reserved
0	GPIO224	R/W	0h	Output Open-Drain control for this pin Reset type: SYSRSn

### 8.10.2.34 GPHAMSEL Register (Offset = 1D4h) [Reset = FFFFFFFFh]

GPHAMSEL is shown in [Figure 8-37](#) and described in [Table 8-45](#).

Return to the [Summary Table](#).

GPIO H Analog Mode Select register (GPIO224 to GPIO255)

Selects between digital and analog functionality for GPIO pins.

0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers

1: The analog function of the pin is enabled and the pin is capable of analog functions

Reading the register returns the current value of the register setting.

Note:

[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, t

**Figure 8-37. GPHAMSEL Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-1h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/W-1h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/W-1h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/W-1h							

**Table 8-45. GPHAMSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	RESERVED	R/W	1h	Reserved
27	RESERVED	R/W	1h	Reserved
26	RESERVED	R/W	1h	Reserved
25	RESERVED	R/W	1h	Reserved
24	RESERVED	R/W	1h	Reserved
23	RESERVED	R/W	1h	Reserved
22	RESERVED	R/W	1h	Reserved
21	GPIO245	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn

**Table 8-45. GPHAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	GPIO244	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
19	RESERVED	R/W	1h	Reserved
18	GPIO242	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
17	GPIO241	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
16	RESERVED	R/W	1h	Reserved
15	GPIO239	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
14	GPIO238	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>

**Table 8-45. GPHAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	GPIO237	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
12	RESERVED	R/W	1h	Reserved
11	RESERVED	R/W	1h	Reserved
10	RESERVED	R/W	1h	Reserved
9	GPIO233	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
8	GPIO232	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
7	GPIO231	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
6	GPIO230	R/W	1h	<p>0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers</p> <p>1: The analog function of the pin is enabled and the pin is capable of analog functions</p> <p>Reading the register returns the current value of the register setting.</p> <p>Note:</p> <p>[1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect.</p> <p>Reset type: SYSRSn</p>
5	RESERVED	R/W	1h	Reserved

**Table 8-45. GPHAMSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	GPIO228	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn
3	GPIO227	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn
2	GPIO226	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn
1	GPIO225	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn
0	GPIO224	R/W	1h	0: The analog function of the pin is disabled and the pin is capable of digital functions as specified by the other GPIO configuration registers 1: The analog function of the pin is enabled and the pin is capable of analog functions Reading the register returns the current value of the register setting. Note: [1] This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. For all the IOs, the corresponding bits in these registers dont have any affect. Reset type: SYSRSn

### 8.10.2.35 GPHGMUX1 Register (Offset = 1E0h) [Reset = 0000000h]

GPHGMUX1 is shown in [Figure 8-38](#) and described in [Table 8-46](#).

Return to the [Summary Table](#).

GPIO H Peripheral Group Mux (GPIO224 to 239)

Defines pin-muxing selection for GPIO.

Notes:

[1]For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-38. GPHGMUX1 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
RESERVED		GPIO230		RESERVED		GPIO228	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
GPIO227		GPIO226		RESERVED		GPIO224	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-46. GPHGMUX1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	GPIO230	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
11-10	RESERVED	R/W	0h	Reserved
9-8	GPIO228	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
7-6	GPIO227	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
5-4	GPIO226	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	RESERVED	R/W	0h	Reserved
1-0	GPIO224	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn

### 8.10.2.36 GPHGMUX2 Register (Offset = 1E2h) [Reset = 0000000h]

GPHGMUX2 is shown in [Figure 8-39](#) and described in [Table 8-47](#).

Return to the [Summary Table](#).

GPIO H Peripheral Group Mux (GPIO240 to 255)

Defines pin-muxing selection for GPIO.

Notes:

[1]For complete pin-mux selection on GPIOx, GPAMUXy.GPIOx configuration is also required.

**Figure 8-39. GPHGMUX2 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
23	22	21	20	19	18	17	16
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
RESERVED		RESERVED		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		GPIO242		RESERVED		RESERVED	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 8-47. GPHGMUX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	RESERVED	R/W	0h	Reserved
27-26	RESERVED	R/W	0h	Reserved
25-24	RESERVED	R/W	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-10	RESERVED	R/W	0h	Reserved
9-8	RESERVED	R/W	0h	Reserved
7-6	RESERVED	R/W	0h	Reserved
5-4	GPIO242	R/W	0h	Defines pin-muxing selection for GPIO Reset type: SYSRSn
3-2	RESERVED	R/W	0h	Reserved
1-0	RESERVED	R/W	0h	Reserved

### 8.10.2.37 GPHLOCK Register (Offset = 1FCh) [Reset = 0000000h]

GPHLOCK is shown in [Figure 8-40](#) and described in [Table 8-48](#).

Return to the [Summary Table](#).

GPIO H Lock Configuration Register (GPIO224 to 255)

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 8-40. GPHLOCK Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/W-0h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/W-0h							

**Table 8-48. GPHLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	GPIO245	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
20	GPIO244	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn

**Table 8-48. GPHLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	RESERVED	R/W	0h	Reserved
18	GPIO242	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
17	GPIO241	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
16	RESERVED	R/W	0h	Reserved
15	GPIO239	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
14	GPIO238	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
13	GPIO237	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	GPIO233	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
8	GPIO232	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn

**Table 8-48. GPHLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	GPIO231	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
6	GPIO230	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
3	GPIO227	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
2	GPIO226	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
1	GPIO225	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn
0	GPIO224	R/W	0h	1: Locks changes to the bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx registers which control the same pin 0: Bits in GPyMUX1, GPyMUX2, GPyDIR, GPyINV, GPyODR, GPyAMSEL, GPyGMUX1, GPyGMUX2 and GPyCSELx register which control the same pin can be changed Reset type: SYSRSn

### 8.10.2.38 GPHCR Register (Offset = 1FEh) [Reset = 0000000h]

GPHCR is shown in [Figure 8-41](#) and described in [Table 8-49](#).

Return to the [Summary Table](#).

GPIO H Lock Commit Register (GPIO224 to 255)

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 8-41. GPHCR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/WOnce-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/WOnce-0h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/WOnce-0h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/WOnce-0h							

**Table 8-49. GPHCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/WOnce	0h	Reserved
30	RESERVED	R/WOnce	0h	Reserved
29	RESERVED	R/WOnce	0h	Reserved
28	RESERVED	R/WOnce	0h	Reserved
27	RESERVED	R/WOnce	0h	Reserved
26	RESERVED	R/WOnce	0h	Reserved
25	RESERVED	R/WOnce	0h	Reserved
24	RESERVED	R/WOnce	0h	Reserved
23	RESERVED	R/WOnce	0h	Reserved
22	RESERVED	R/WOnce	0h	Reserved
21	GPIO245	R/WOnce	0h	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 Reset type: SYSRSn
20	GPIO244	R/WOnce	0h	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 Reset type: SYSRSn
19	RESERVED	R/WOnce	0h	Reserved
18	GPIO242	R/WOnce	0h	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 Reset type: SYSRSn

**Table 8-49. GPHCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO241	R/WOnce	0h	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 Reset type: SYSRSn
16	RESERVED	R/WOnce	0h	Reserved
15	GPIO239	R/WOnce	0h	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 Reset type: SYSRSn
14	GPIO238	R/WOnce	0h	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 Reset type: SYSRSn
13	GPIO237	R/WOnce	0h	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 Reset type: SYSRSn
12	RESERVED	R/WOnce	0h	Reserved
11	RESERVED	R/WOnce	0h	Reserved
10	RESERVED	R/WOnce	0h	Reserved
9	GPIO233	R/WOnce	0h	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 Reset type: SYSRSn
8	GPIO232	R/WOnce	0h	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 Reset type: SYSRSn
7	GPIO231	R/WOnce	0h	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 Reset type: SYSRSn
6	GPIO230	R/WOnce	0h	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 Reset type: SYSRSn
5	RESERVED	R/WOnce	0h	Reserved
4	GPIO228	R/WOnce	0h	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 Reset type: SYSRSn
3	GPIO227	R/WOnce	0h	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 Reset type: SYSRSn

**Table 8-49. GPHCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	GPIO226	R/WOnce	0h	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 Reset type: SYSRSn
1	GPIO225	R/WOnce	0h	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 Reset type: SYSRSn
0	GPIO224	R/WOnce	0h	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 Reset type: SYSRSn

### 8.10.3 GPIO\_DATA\_REGS Registers

Table 8-50 lists the memory-mapped registers for the GPIO\_DATA\_REGS registers. All register offset addresses not listed in Table 8-50 should be considered as reserved locations and the register contents should not be modified.

**Table 8-50. GPIO\_DATA\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GPADAT	GPIO A Data Register (GPIO0 to 31)		<a href="#">Go</a>
2h	GPASET	GPIO A Data Set Register (GPIO0 to 31)		<a href="#">Go</a>
4h	GPACLEAR	GPIO A Data Clear Register (GPIO0 to 31)		<a href="#">Go</a>
6h	GPATOGGLE	GPIO A Data Toggle Register (GPIO0 to 31)		<a href="#">Go</a>
8h	GPBDAT	GPIO B Data Register (GPIO32 to 63)		<a href="#">Go</a>
Ah	GPBSET	GPIO B Data Set Register (GPIO32 to 63)		<a href="#">Go</a>
Ch	GPBCLEAR	GPIO B Data Clear Register (GPIO32 to 63)		<a href="#">Go</a>
Eh	GPBTOGGLE	GPIO B Data Toggle Register (GPIO32 to 63)		<a href="#">Go</a>
38h	GPHDAT	GPIO H Data Register (GPIO224 to 255)		<a href="#">Go</a>
3Ah	GPHSET	GPIO H Data Set Register (GPIO224 to 255)		<a href="#">Go</a>
3Ch	GPHCLEAR	GPIO H Data Clear Register (GPIO224 to 255)		<a href="#">Go</a>
3Eh	GPHTOGGLE	GPIO H Data Toggle Register (GPIO224 to 255)		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 8-51 shows the codes that are used for access types in this section.

**Table 8-51. GPIO\_DATA\_REGS Access Type Codes**

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

### 8.10.3.1 GPADAT Register (Offset = 0h) [Reset = 0000000h]

GPADAT is shown in [Figure 8-42](#) and described in [Table 8-52](#).

Return to the [Summary Table](#).

GPIO A Data Register (GPIO0 to 31)

Reading this register indicates the current status of the GPIO pin, irrespective of which mode the pin is in. Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.

DESIGNER NOTE:

[1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.

**Figure 8-42. GPADAT Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	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
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 8-52. GPADAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	GPIO29	R/W	0h	Data Register for this pin Reset type: SYSRSn
28	GPIO28	R/W	0h	Data Register for this pin Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	GPIO24	R/W	0h	Data Register for this pin Reset type: SYSRSn
23	GPIO23	R/W	0h	Data Register for this pin Reset type: SYSRSn
22	GPIO22	R/W	0h	Data Register for this pin Reset type: SYSRSn
21	GPIO21	R/W	0h	Data Register for this pin Reset type: SYSRSn
20	GPIO20	R/W	0h	Data Register for this pin Reset type: SYSRSn

**Table 8-52. GPADAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	GPIO19	R/W	0h	Data Register for this pin Reset type: SYSRSn
18	GPIO18	R/W	0h	Data Register for this pin Reset type: SYSRSn
17	GPIO17	R/W	0h	Data Register for this pin Reset type: SYSRSn
16	GPIO16	R/W	0h	Data Register for this pin Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	Data Register for this pin Reset type: SYSRSn
12	GPIO12	R/W	0h	Data Register for this pin Reset type: SYSRSn
11	GPIO11	R/W	0h	Data Register for this pin Reset type: SYSRSn
10	GPIO10	R/W	0h	Data Register for this pin Reset type: SYSRSn
9	GPIO9	R/W	0h	Data Register for this pin Reset type: SYSRSn
8	GPIO8	R/W	0h	Data Register for this pin Reset type: SYSRSn
7	GPIO7	R/W	0h	Data Register for this pin Reset type: SYSRSn
6	GPIO6	R/W	0h	Data Register for this pin Reset type: SYSRSn
5	GPIO5	R/W	0h	Data Register for this pin Reset type: SYSRSn
4	GPIO4	R/W	0h	Data Register for this pin Reset type: SYSRSn
3	GPIO3	R/W	0h	Data Register for this pin Reset type: SYSRSn
2	GPIO2	R/W	0h	Data Register for this pin Reset type: SYSRSn
1	GPIO1	R/W	0h	Data Register for this pin Reset type: SYSRSn
0	GPIO0	R/W	0h	Data Register for this pin Reset type: SYSRSn

### 8.10.3.2 GPASET Register (Offset = 2h) [Reset = 0000000h]

GPASET is shown in [Figure 8-43](#) and described in [Table 8-53](#).

Return to the [Summary Table](#).

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 8-43. GPASET Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	GPIO24
R-0/W-0h							
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R-0/W-0h							

**Table 8-53. GPASET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	GPIO29	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
28	GPIO28	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	GPIO24	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
23	GPIO23	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
22	GPIO22	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
21	GPIO21	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
20	GPIO20	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
19	GPIO19	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
18	GPIO18	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn

**Table 8-53. GPASET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
16	GPIO16	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	GPIO13	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
12	GPIO12	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
11	GPIO11	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
10	GPIO10	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
9	GPIO9	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
8	GPIO8	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
7	GPIO7	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
6	GPIO6	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
5	GPIO5	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
4	GPIO4	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
3	GPIO3	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
2	GPIO2	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
1	GPIO1	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
0	GPIO0	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn

### 8.10.3.3 GPACLEAR Register (Offset = 4h) [Reset = 0000000h]

GPACLEAR is shown in [Figure 8-44](#) and described in [Table 8-54](#).

Return to the [Summary Table](#).

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 8-44. GPACLEAR Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	GPIO24
R-0/W-0h							
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R-0/W-0h							

**Table 8-54. GPACLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	GPIO29	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
28	GPIO28	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	GPIO24	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
23	GPIO23	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
22	GPIO22	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
21	GPIO21	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
20	GPIO20	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
19	GPIO19	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
18	GPIO18	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn

**Table 8-54. GPACLEAR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
16	GPIO16	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	GPIO13	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
12	GPIO12	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
11	GPIO11	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
10	GPIO10	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
9	GPIO9	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
8	GPIO8	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
7	GPIO7	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
6	GPIO6	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
5	GPIO5	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
4	GPIO4	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
3	GPIO3	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
2	GPIO2	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
1	GPIO1	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
0	GPIO0	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn

### 8.10.3.4 GPATOGGLE Register (Offset = 6h) [Reset = 0000000h]

GPATOGGLE is shown in [Figure 8-45](#) and described in [Table 8-55](#).

Return to the [Summary Table](#).

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 8-45. GPATOGGLE Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	GPIO29	GPIO28	RESERVED	RESERVED	RESERVED	GPIO24
R-0/W-0h							
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R-0/W-0h							

**Table 8-55. GPATOGGLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	GPIO29	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
28	GPIO28	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	GPIO24	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
23	GPIO23	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
22	GPIO22	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
21	GPIO21	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
20	GPIO20	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
19	GPIO19	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
18	GPIO18	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn

**Table 8-55. GPATOGGLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	GPIO17	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
16	GPIO16	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	GPIO13	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
12	GPIO12	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
11	GPIO11	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
10	GPIO10	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
9	GPIO9	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
8	GPIO8	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
7	GPIO7	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
6	GPIO6	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
5	GPIO5	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
4	GPIO4	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
3	GPIO3	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
2	GPIO2	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
1	GPIO1	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
0	GPIO0	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn

### 8.10.3.5 GPBDAT Register (Offset = 8h) [Reset = 0000000h]

GPBDAT is shown in [Figure 8-46](#) and described in [Table 8-56](#).

Return to the [Summary Table](#).

GPIO B Data Register (GPIO32 to 63)

Reading this register indicates the current status of the GPIO pin, irrespective of which mode the pin is in. Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.

DESIGNER NOTE:

[1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.

**Figure 8-46. GPBDAT Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R/W-0h							

**Table 8-56. GPBDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved

**Table 8-56. GPBDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	GPIO41	R/W	0h	Data Register for this pin Reset type: SYSRSn
8	GPIO40	R/W	0h	Data Register for this pin Reset type: SYSRSn
7	GPIO39	R/W	0h	Data Register for this pin Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	GPIO37	R/W	0h	Data Register for this pin Reset type: SYSRSn
4	RESERVED	R/W	0h	Reserved
3	GPIO35	R/W	0h	Data Register for this pin Reset type: SYSRSn
2	RESERVED	R/W	0h	Reserved
1	GPIO33	R/W	0h	Data Register for this pin Reset type: SYSRSn
0	GPIO32	R/W	0h	Data Register for this pin Reset type: SYSRSn

### 8.10.3.6 GPBSET Register (Offset = Ah) [Reset = 0000000h]

GPBSET is shown in [Figure 8-47](#) and described in [Table 8-57](#).

Return to the [Summary Table](#).

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 8-47. GPBSET Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R-0/W-0h							

**Table 8-57. GPBSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	RESERVED	R-0/W	0h	Reserved
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-57. GPBSET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
8	GPIO40	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
7	GPIO39	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
6	RESERVED	R-0/W	0h	Reserved
5	GPIO37	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
4	RESERVED	R-0/W	0h	Reserved
3	GPIO35	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
2	RESERVED	R-0/W	0h	Reserved
1	GPIO33	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
0	GPIO32	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn

### 8.10.3.7 GPBCLEAR Register (Offset = Ch) [Reset = 0000000h]

GPBCLEAR is shown in [Figure 8-48](#) and described in [Table 8-58](#).

Return to the [Summary Table](#).

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 8-48. GPBCLEAR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R-0/W-0h							

**Table 8-58. GPBCLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	RESERVED	R-0/W	0h	Reserved
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-58. GPBCLEAR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
8	GPIO40	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
7	GPIO39	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
6	RESERVED	R-0/W	0h	Reserved
5	GPIO37	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
4	RESERVED	R-0/W	0h	Reserved
3	GPIO35	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
2	RESERVED	R-0/W	0h	Reserved
1	GPIO33	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
0	GPIO32	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn

### 8.10.3.8 GPBTOGGLE Register (Offset = Eh) [Reset = 0000000h]

GPBTOGGLE is shown in [Figure 8-49](#) and described in [Table 8-59](#).

Return to the [Summary Table](#).

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 8-49. GPBTOGGLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO41	GPIO40
R-0/W-0h							
7	6	5	4	3	2	1	0
GPIO39	RESERVED	GPIO37	RESERVED	GPIO35	RESERVED	GPIO33	GPIO32
R-0/W-0h							

**Table 8-59. GPBTOGGLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	RESERVED	R-0/W	0h	Reserved
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-59. GPBTOGGLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	GPIO41	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
8	GPIO40	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
7	GPIO39	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
6	RESERVED	R-0/W	0h	Reserved
5	GPIO37	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
4	RESERVED	R-0/W	0h	Reserved
3	GPIO35	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
2	RESERVED	R-0/W	0h	Reserved
1	GPIO33	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
0	GPIO32	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn

### 8.10.3.9 GPHDAT Register (Offset = 38h) [Reset = 0000000h]

GPHDAT is shown in [Figure 8-50](#) and described in [Table 8-60](#).

Return to the [Summary Table](#).

GPIO H Data Register (GPIO224 to 255)

Reading this register indicates the current status of the GPIO pin, irrespective of which mode the pin is in. Writing to this register will set the GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.

DESIGNER NOTE:

[1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.

**Figure 8-50. GPHDAT Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO245	GPIO244	RESERVED	GPIO242	GPIO241	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
GPIO239	GPIO238	GPIO237	RESERVED	RESERVED	RESERVED	GPIO233	GPIO232
R/W-0h							
7	6	5	4	3	2	1	0
GPIO231	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	GPIO225	GPIO224
R/W-0h							

**Table 8-60. GPHDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	GPIO245	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn

**Table 8-60. GPHDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20	GPIO244	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
19	RESERVED	R/W	0h	Reserved
18	GPIO242	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
17	GPIO241	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
16	RESERVED	R/W	0h	Reserved
15	GPIO239	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn

**Table 8-60. GPHDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	GPIO238	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
13	GPIO237	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	GPIO233	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
8	GPIO232	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn

**Table 8-60. GPHDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	GPIO231	R/W	0h	<p>Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.</p> <p>DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.</p> <p>Reset type: SYSRSn</p>
6	GPIO230	R/W	0h	<p>Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.</p> <p>DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.</p> <p>Reset type: SYSRSn</p>
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	<p>Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.</p> <p>DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.</p> <p>Reset type: SYSRSn</p>
3	GPIO227	R/W	0h	<p>Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.</p> <p>DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.</p> <p>Reset type: SYSRSn</p>
2	GPIO226	R/W	0h	<p>Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero.</p> <p>DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register.</p> <p>Reset type: SYSRSn</p>

**Table 8-60. GPHDAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	GPIO225	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn
0	GPIO224	R/W	0h	Reading this register indicates the current status of this GPIO pin, irrespective of which mode the pin is in. Writing to this register will set this GPIO pin high or low if the pin is enabled for GPIO output mode, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A system reset will clear all bits and latched values to zero. DESIGNER NOTE: [1] Reading the GPIODAT register should reflect the state of the PIN (after qualification), not the state of the output latch of the GPIODAT register. Reset type: SYSRSn

### 8.10.3.10 GPHSET Register (Offset = 3Ah) [Reset = 0000000h]

GPHSET is shown in [Figure 8-51](#) and described in [Table 8-61](#).

Return to the [Summary Table](#).

GPIO H Data Set Register (GPIO224 to 255)

Writing a 1 will force GPIO output data latch to 1.

Writes of 0 are ignored.

Always reads back a 0.

**Figure 8-51. GPHSET Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R-0/W-0h							

**Table 8-61. GPHSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	GPIO242	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-61. GPHSET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R-0/W	0h	Reserved
8	RESERVED	R-0/W	0h	Reserved
7	RESERVED	R-0/W	0h	Reserved
6	GPIO230	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
5	RESERVED	R-0/W	0h	Reserved
4	GPIO228	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
3	GPIO227	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
2	GPIO226	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn
1	RESERVED	R-0/W	0h	Reserved
0	GPIO224	R-0/W	0h	Output Set bit for this pin Reset type: SYSRSn

### 8.10.3.11 GPHCLEAR Register (Offset = 3Ch) [Reset = 0000000h]

GPHCLEAR is shown in [Figure 8-52](#) and described in [Table 8-62](#).

Return to the [Summary Table](#).

GPIO H Data Clear Register (GPIO224 to 255)

Writing a 1 will force GPIO0 output data latch to 0.

Writes of 0 are ignored.

Always reads back a 0.

**Figure 8-52. GPHCLEAR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R-0/W-0h							

**Table 8-62. GPHCLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	GPIO242	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-62. GPHCLEAR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R-0/W	0h	Reserved
8	RESERVED	R-0/W	0h	Reserved
7	RESERVED	R-0/W	0h	Reserved
6	GPIO230	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
5	RESERVED	R-0/W	0h	Reserved
4	GPIO228	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
3	GPIO227	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
2	GPIO226	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn
1	RESERVED	R-0/W	0h	Reserved
0	GPIO224	R-0/W	0h	Output Clear bit for this pin Reset type: SYSRSn

### 8.10.3.12 GPHTOGGLE Register (Offset = 3Eh) [Reset = 0000000h]

GPHTOGGLE is shown in [Figure 8-53](#) and described in [Table 8-63](#).

Return to the [Summary Table](#).

GPIO H Data Toggle Register (GPIO224 to 255)

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 8-53. GPHTOGGLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R-0/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R-0/W-0h							

**Table 8-63. GPHTOGGLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W	0h	Reserved
30	RESERVED	R-0/W	0h	Reserved
29	RESERVED	R-0/W	0h	Reserved
28	RESERVED	R-0/W	0h	Reserved
27	RESERVED	R-0/W	0h	Reserved
26	RESERVED	R-0/W	0h	Reserved
25	RESERVED	R-0/W	0h	Reserved
24	RESERVED	R-0/W	0h	Reserved
23	RESERVED	R-0/W	0h	Reserved
22	RESERVED	R-0/W	0h	Reserved
21	RESERVED	R-0/W	0h	Reserved
20	RESERVED	R-0/W	0h	Reserved
19	RESERVED	R-0/W	0h	Reserved
18	GPIO242	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
17	RESERVED	R-0/W	0h	Reserved
16	RESERVED	R-0/W	0h	Reserved
15	RESERVED	R-0/W	0h	Reserved
14	RESERVED	R-0/W	0h	Reserved
13	RESERVED	R-0/W	0h	Reserved
12	RESERVED	R-0/W	0h	Reserved
11	RESERVED	R-0/W	0h	Reserved
10	RESERVED	R-0/W	0h	Reserved

**Table 8-63. GPHTOGGLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R-0/W	0h	Reserved
8	RESERVED	R-0/W	0h	Reserved
7	RESERVED	R-0/W	0h	Reserved
6	GPIO230	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
5	RESERVED	R-0/W	0h	Reserved
4	GPIO228	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
3	GPIO227	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
2	GPIO226	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn
1	RESERVED	R-0/W	0h	Reserved
0	GPIO224	R-0/W	0h	Output Toggle Register GPIO pin Reset type: SYSRSn

### 8.10.4 GPIO\_DATA\_READ\_REGS Registers

Table 8-64 lists the memory-mapped registers for the GPIO\_DATA\_READ\_REGS registers. All register offset addresses not listed in Table 8-64 should be considered as reserved locations and the register contents should not be modified.

**Table 8-64. GPIO\_DATA\_READ\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GPADAT_R	GPIO A Data Read Register		<a href="#">Go</a>
2h	GPBDAT_R	GPIO B Data Read Register		<a href="#">Go</a>
Eh	GPHDAT_R	GPIO H Data Read Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 8-65 shows the codes that are used for access types in this section.

**Table 8-65. GPIO\_DATA\_READ\_REGS Access Type Codes**

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

### 8.10.4.1 GPADAT\_R Register (Offset = 0h) [Reset = 00000000h]

GPADAT\_R is shown in [Figure 8-54](#) and described in [Table 8-66](#).

Return to the [Summary Table](#).

GPIO A Data Read Register.

Returns the contents of GPADAT register on a read, write to this register has no effect

**Figure 8-54. GPADAT\_R Register**

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

**Table 8-66. GPADAT\_R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Returns the contents of what was written to the GPADAT register on a read, write to this register has no effect Reset type: CPU1.SYSRSn

### 8.10.4.2 GPBDAT\_R Register (Offset = 2h) [Reset = 0000000h]

GPBDAT\_R is shown in [Figure 8-55](#) and described in [Table 8-67](#).

Return to the [Summary Table](#).

GPIO B Data Read Register.

Returns the contents of GPBDAT register on a read, write to this register has no effect

**Figure 8-55. GPBDAT\_R Register**

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

**Table 8-67. GPBDAT\_R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Returns the contents of what was written to the GPBDAT register on a read, write to this register has no effect Reset type: CPU1.SYSRSn

### 8.10.4.3 GPHDAT\_R Register (Offset = Eh) [Reset = 0000000h]

GPHDAT\_R is shown in [Figure 8-56](#) and described in [Table 8-68](#).

Return to the [Summary Table](#).

GPIO H Data Read Register.

Returns the contents of GPHDAT register on a read, write to this register has no effect

**Figure 8-56. GPHDAT\_R Register**

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

**Table 8-68. GPHDAT\_R Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DATA	R	0h	Returns the contents of what was written to the GPHDAT register on a read, write to this register has no effect Reset type: CPU1.SYSRSn

### 8.10.5 GPIO Registers to Driverlib Functions

**Table 8-69. GPIO Registers to Driverlib Functions**

File	Driverlib Function
<b>GPACTRL</b>	
gpio.c	GPIO_setQualificationPeriod
<b>GPAQSEL1</b>	
gpio.c	GPIO_setQualificationMode
gpio.c	GPIO_getQualificationMode
<b>GPAQSEL2</b>	
-	See GPAQSEL1
<b>GPAMUX1</b>	
gpio.c	GPIO_setPinConfig
<b>GPAMUX2</b>	
-	See GPAMUX1
<b>GPADIR</b>	
gpio.c	GPIO_setDirectionMode
gpio.c	GPIO_getDirectionMode
<b>GPAPUD</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAINV</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAODR</b>	
gpio.c	GPIO_setPadConfig
gpio.c	GPIO_getPadConfig
<b>GPAAMSEL</b>	
-	
<b>GPAGMUX1</b>	
gpio.c	GPIO_setPinConfig
<b>GPAGMUX2</b>	

**Table 8-69. GPIO Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See GPAGMUX1
<b>GPALOCK</b>	
gpio.h	GPIO_lockPortConfig
gpio.h	GPIO_unlockPortConfig
<b>GPACR</b>	
gpio.h	GPIO_commitPortConfig
<b>GPBCTRL</b>	
-	See GPECTRL
<b>GPBQSEL1</b>	
-	See GPAQSEL1
<b>GPBMUX1</b>	
-	See GPAMUX1
<b>GPBDIR</b>	
-	See GPADIR
<b>GPBPUD</b>	
-	See GPAPUD
<b>GPBINV</b>	
-	See GPAINV
<b>GPBODR</b>	
-	See GPAODR
<b>GPBGMUX1</b>	
-	See GPAGMUX1
<b>GPBLOCK</b>	
-	See GPALOCK
<b>GPBCR</b>	
-	See GPACR
<b>GPHCTRL</b>	
-	See GPECTRL
<b>GPHQSEL1</b>	
-	See GPAQSEL1
<b>GPHQSEL2</b>	
-	See GPAQSEL1
<b>GPHMUX1</b>	
-	See GPAMUX1
<b>GPHMUX2</b>	
-	See GPAMUX1
<b>GPHDIR</b>	
-	See GPADIR
<b>GPHPUD</b>	
-	See GPAPUD
<b>GPHINV</b>	
-	See GPAINV
<b>GPHODR</b>	
-	See GPAODR
<b>GPHAMSEL</b>	

**Table 8-69. GPIO Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	
<b>GPHGMUX1</b>	
-	See GPAGMUX1
<b>GPHGMUX2</b>	
-	See GPAGMUX1
<b>GPHLOCK</b>	
-	See GPALOCK
<b>GPHCR</b>	
-	See GPACR
<b>GPADAT</b>	
gpio.h	GPIO_readPin
gpio.h	GPIO_readPortData
gpio.h	GPIO_writePortData
<b>GPASET</b>	
gpio.h	GPIO_writePin
gpio.h	GPIO_setPortPins
<b>GPACLEAR</b>	
gpio.h	GPIO_writePin
gpio.h	GPIO_clearPortPins
<b>GPATOGGLE</b>	
gpio.h	GPIO_togglePin
gpio.h	GPIO_togglePortPins
<b>GPBDAT</b>	
-	See GPADAT
<b>GPBSET</b>	
-	See GPASET
<b>GPBCLEAR</b>	
-	See GPACLEAR
<b>GPBTOGGLE</b>	
-	See GPATOGGLE
<b>GPHDAT</b>	
-	See GPADAT
<b>GPHSET</b>	
-	See GPASET
<b>GPHCLEAR</b>	
-	See GPACLEAR
<b>GPHTOGGLE</b>	
-	See GPATOGGLE
<b>GPADAT_R</b>	
-	
<b>GPBDAT_R</b>	
-	
<b>GPHDAT_R</b>	
-	



The crossbars (referred to as X-BAR throughout this chapter) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations.

The device contains a total of three X-BARs:

- Input X-BAR
- Output X-BAR
- ePWM X-BAR

Each of the X-BARs is named according to where the X-BAR takes signals. For example, the Input X-BAR brings external signals “in” to the device. The Output X-BAR takes internal signals “out” of the device to a GPIO. The ePWM X-BAR takes signals to the ePWM modules.

<b>9.1 Input X-BAR</b> .....	<a href="#">724</a>
<b>9.2 ePWM and GPIO Output X-BAR</b> .....	<a href="#">727</a>
<b>9.3 XBAR Registers</b> .....	<a href="#">732</a>

## 9.1 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, eCAP, ePWM, and external interrupts. The input of each Input X-BAR instance (INPUTx) can be any GPIO, while the output of each instance connects to various IP blocks in the device. The digital input of AIOs are also available as inputs to the Input X-BAR. This flexibility relieves some of the constraints on peripheral muxing by allowing the user to connect any GPIO to the specified outputs of each Input X-BAR instance. Note that the GPIO selected by the Input X-BAR can be configured as either an input or an output. The Input X-BAR simply connects the signal on the input buffer to the output of the selected Input X-BAR instance. Therefore, you can do things such as route the output of an ePWM to the eCAP module for a frequency test).

The Input X-BAR is configured by way of the INPUTxSELECT registers. The destinations for each INPUTx are shown in [Figure 9-1](#) and [Table 9-1](#). For additional details on how each Input X-BAR connects to other IP blocks throughout the device, look for references to Input X-BAR in the chapter associated with that IP. Note that the destinations of each INPUTx are fixed and are not user-configurable. For more information on configuring the Input X-BAR, see the INPUT\_XBAR\_REGS register definitions in the *XBAR Registers* section.

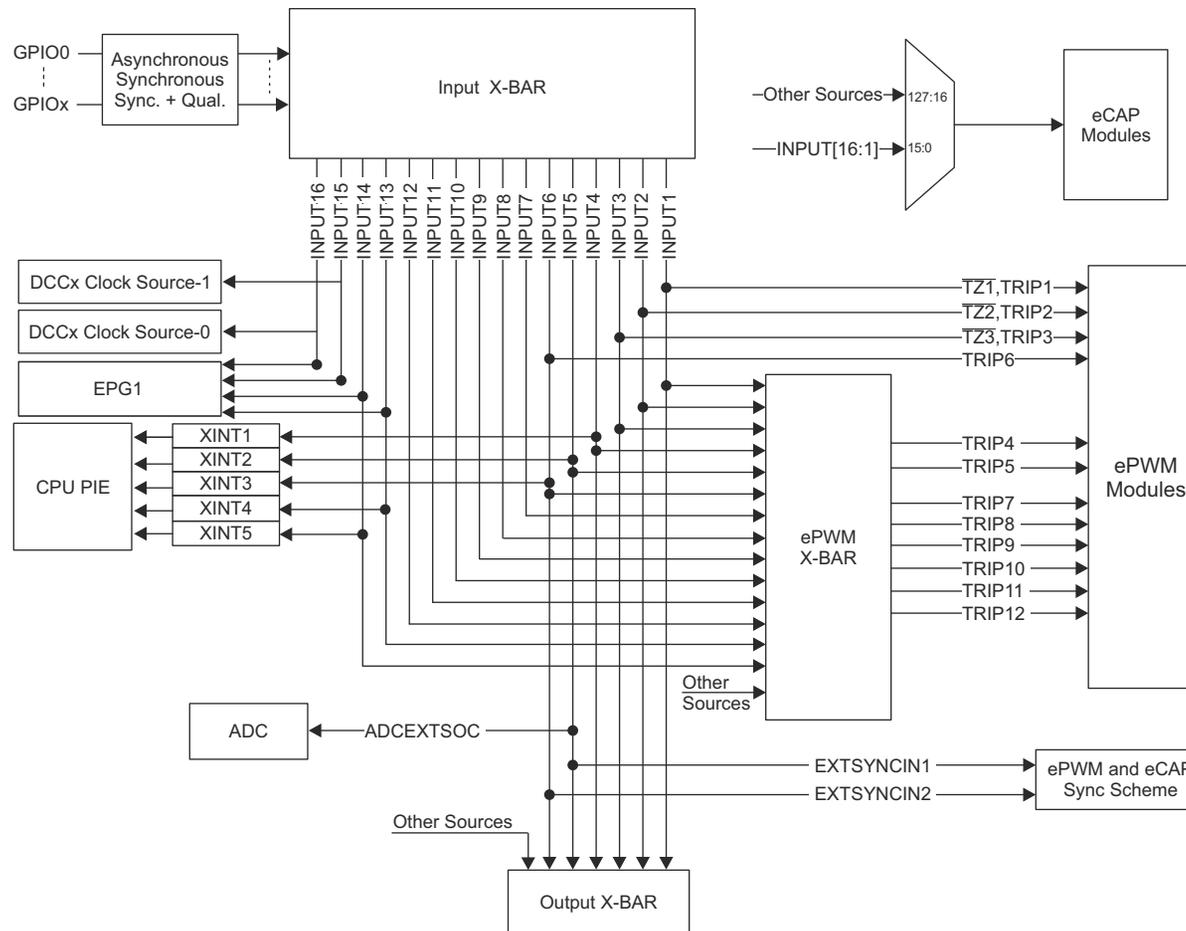


Figure 9-1. Input X-BAR

**Note**

INPUTXBARx, INPUTXBAR\_INPUTx, and INPUTx (when referenced in the context of Input X-BAR) are equivalent in all C2000 software and documentation.

**Table 9-1. Input X-BAR Destinations**

INPUT	ECAP	EPWM X-BAR	OUTPUT X-BAR	CPU XINT	EPWM TRIP	ADC START OF CONVERSION	EPWM / ECAP SYNC	DCCx	EPG
1	Yes	Yes	Yes	-	TZ1,TRIP1	-	-	-	-
2	Yes	Yes	Yes	-	TZ2,TRIP2	-	-	-	-
3	Yes	Yes	Yes	-	TZ3,TRIP3	-	-	-	-
4	Yes	Yes	Yes	XINT1	-	-	-	-	-
5	Yes	Yes	Yes	XINT2	-	ADCEXTSOC	EXTSYNCIN1	-	-
6	Yes	Yes	Yes	XINT3	TRIP6	-	EXTSYNCIN2	-	-
7	Yes	Yes	-	-	-	-	-	-	-
8	Yes	Yes	-	-	-	-	-	-	-
9	Yes	Yes	-	-	-	-	-	-	-
10	Yes	Yes	-	-	-	-	-	-	-
11	Yes	Yes	-	-	-	-	-	CLK1	-
12	Yes	Yes	-	-	-	-	-	CLK1	-
13	Yes	Yes	-	XINT4	-	-	-	-	EPG1IN1
14	Yes	Yes	-	XINT5	-	-	-	-	EPG1IN2
15	Yes	-	-	-	-	-	-	CLK1	EPG1IN3
16	Yes	-	-	-	-	-	-	CLK0	EPG1IN4

## 9.2 ePWM and GPIO Output X-BAR

This section describes the ePWM and GPIO Output X-BAR. .

### 9.2.1 ePWM X-BAR

The ePWM X-BAR brings signals to the ePWM modules. Specifically, the ePWM X-BAR is connected to the Digital Compare (DC) submodule of each ePWM module for actions such as tripzones and syncing. Refer to the *Enhanced Pulse Width Modulator (ePWM)* chapter for more information on additional ways the DC submodule can be used. Figure 9-2 shows the architecture of the ePWM X-BAR. Note that the architecture of the ePWM X-BAR is identical to the architecture of the GPIO Output X-BAR (with the exception of the output latch).

#### 9.2.1.1 ePWM X-BAR Architecture

The ePWM X-BAR has eight outputs that are routed to each ePWM module. Figure 9-2 represents the architecture of a single output, but this output is identical to the architecture of all of the other outputs.

First, determine the signals that can be passed to the ePWM by referencing Table 9-2. Select up to one signal per mux for each TRIPx output. Select the inputs to ePWM X-BAR using the TRIPxMUX0TO15CFG and TRIPxMUX16TO31CFG registers. To pass any signal through to the ePWM, enable the signal using the TRIPxMUXENABLE register. All signals that are enabled are logically ORed before being passed on to the respective TRIPx signal on the ePWM. To optionally invert the signal, use the TRIPxOUTINV register.

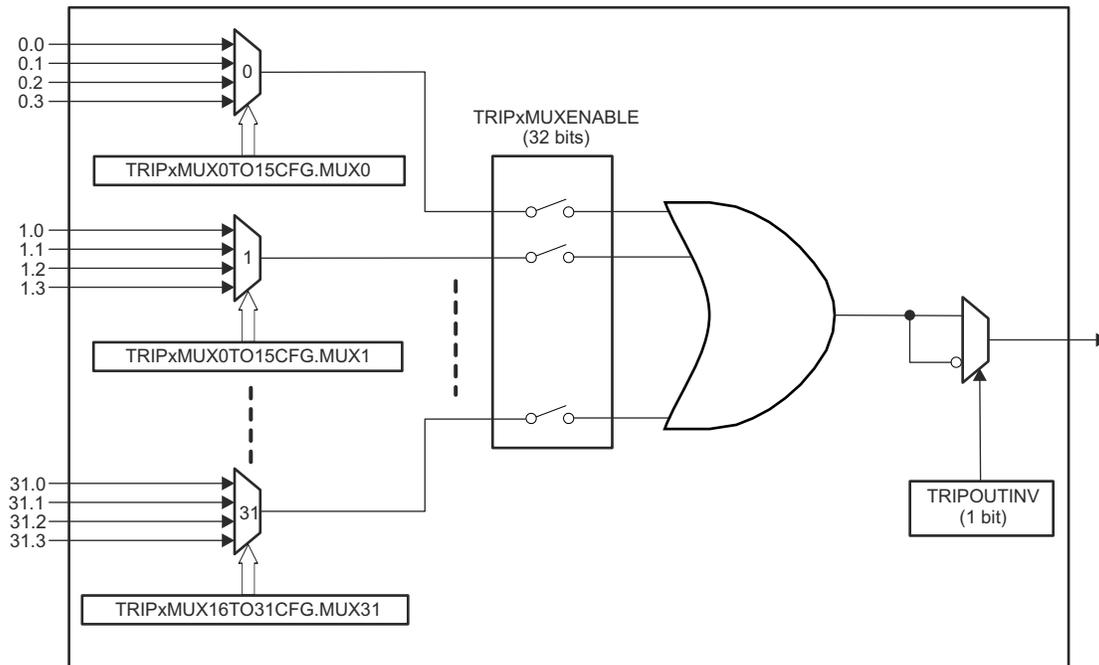


Figure 9-2. ePWM X-BAR Architecture - Single Output

**Note**

Do not use "Reserved" signals in your application.

**Table 9-2. EPWM X-BAR Mux Configuration Table**

Mux	0	1	2	3
G0	CMPSS1_CTRIPH	CMPSS1_CTRIPH_OR_CTRIPL	ADCAEVT1	ECAP1_OUT
G1	CMPSS1_CTRIPL	INPUTXBAR1	Reserved	ADCCEVT1
G2	CMPSS2_CTRIPH	CMPSS2_CTRIPH_OR_CTRIPL	ADCAEVT2	ECAP2_OUT
G3	CMPSS2_CTRIPL	INPUTXBAR2	Reserved	ADCCEVT2
G4	CMPSS3_CTRIPH	CMPSS3_CTRIPH_OR_CTRIPL	ADCAEVT3	Reserved
G5	CMPSS3_CTRIPL	INPUTXBAR3	Reserved	ADCCEVT3
G6	CMPSS4_CTRIPH	CMPSS4_CTRIPH_OR_CTRIPL	ADCAEVT4	Reserved
G7	CMPSS4_CTRIPL	INPUTXBAR4	Reserved	ADCCEVT4
G8	Reserved	Reserved	Reserved	Reserved
G9	Reserved	INPUTXBAR5	Reserved	Reserved
G10	Reserved	Reserved	Reserved	Reserved
G11	Reserved	INPUTXBAR6	Reserved	Reserved
G12	Reserved	Reserved	Reserved	Reserved
G13	Reserved	ADCSOCAO	Reserved	Reserved
G14	Reserved	Reserved	Reserved	EXTSYNCOU
G15	Reserved	ADCSOCBO	Reserved	Reserved
G16	Reserved	Reserved	Reserved	Reserved
G17	Reserved	INPUTXBAR7	Reserved	Reserved
G18	Reserved	Reserved	Reserved	Reserved
G19	Reserved	INPUTXBAR8	Reserved	ERRORSTS
G20	Reserved	Reserved	Reserved	Reserved
G21	Reserved	INPUTXBAR9	Reserved	Reserved
G22	Reserved	Reserved	Reserved	Reserved
G23	Reserved	INPUTXBAR10	Reserved	Reserved
G24	Reserved	Reserved	Reserved	Reserved
G25	Reserved	INPUTXBAR11	Reserved	Reserved
G26	Reserved	Reserved	Reserved	Reserved
G27	Reserved	INPUTXBAR12	Reserved	Reserved
G28	Reserved	Reserved	Reserved	Reserved
G29	Reserved	INPUTXBAR13	Reserved	Reserved
G30	Reserved	Reserved	Reserved	Reserved
G31	Reserved	INPUTXBAR14	ERRORSTS	Reserved

## 9.2.2 GPIO Output X-BAR

The GPIO Output X-BAR takes signals from inside the device and brings them out to a GPIO. Figure 9-3 shows the architecture of the GPIO Output X-BAR. The X-BAR contains eight outputs and each contains at least one position on the GPIO mux, denoted as OUTPUTXBARx. The X-BAR allows the selection of a single signal or a logical-OR of up to 32 signals.

### 9.2.2.1 GPIO Output X-BAR Architecture

The GPIO Output X-BAR has eight outputs that are routed to the GPIO module. Figure 9-3 represents the architecture of a single output, but this output is identical to the architecture of all of the other outputs. Note that the architecture of the Output X-BAR (with the exception of the output latch) is similar to the architecture of the ePWM X-BAR.

First, determine the signals that can be passed to the GPIO by referencing Table 9-3. Select up to one signal per mux (32 total muxes) for each OUTPUTXBARx output. Select the inputs to each mux using the OUTPUTxMUX0TO15CFG and OUTPUTxMUX16TO31CFG registers. To pass any signal through to the GPIO, enable the mux in the OUTPUTxMUXENABLE register. All muxes that are enabled are logically ORed before being passed on to the respective OUTPUTx signal on the GPIO module. To optionally invert the signal, use the OUTPUTINV register. The signal is only recognized on the GPIO, if the proper OUTPUTx muxing options are selected using the GpioCtrlRegs.GPxMUX and GpioCtrlRegs.GPxGMUX registers.

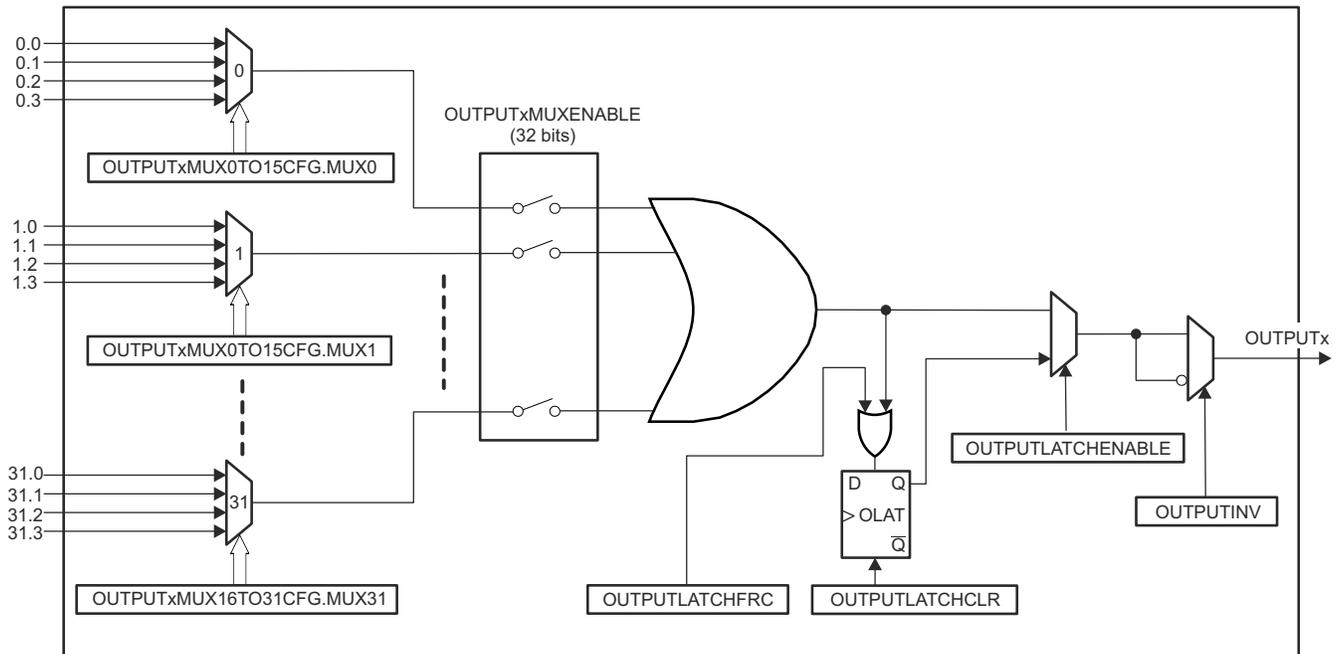


Figure 9-3. GPIO Output X-BAR Architecture

#### Note

Do not use "Reserved" signals in your application.

**Table 9-3. Output X-BAR Mux Configuration Table**

Mux	0	1	2	3
G0	CMPSS1_CTRIPOUTH	CMPSS1_CTRIPOUTH_OR_CTRI POUTL	ADCAEVT1	ECAP1_OUT
G1	CMPSS1_CTRIPOUTL	INPUTXBAR1	Reserved	ADCCEVT1
G2	CMPSS2_CTRIPOUTH	CMPSS2_CTRIPOUTH_OR_CTRI POUTL	ADCAEVT2	ECAP2_OUT
G3	CMPSS2_CTRIPOUTL	INPUTXBAR2	Reserved	ADCCEVT2
G4	CMPSS3_CTRIPOUTH	CMPSS3_CTRIPOUTH_OR_CTRI POUTL	ADCAEVT3	Reserved
G5	CMPSS3_CTRIPOUTL	INPUTXBAR3	Reserved	ADCCEVT3
G6	CMPSS4_CTRIPOUTH	CMPSS4_CTRIPOUTH_OR_CTRI POUTL	ADCAEVT4	Reserved
G7	CMPSS4_CTRIPOUTL	INPUTXBAR4	Reserved	ADCCEVT4
G8	Reserved	Reserved	Reserved	Reserved
G9	Reserved	INPUTXBAR5	Reserved	Reserved
G10	Reserved	Reserved	Reserved	Reserved
G11	Reserved	INPUTXBAR6	Reserved	Reserved
G12	Reserved	Reserved	Reserved	Reserved
G13	Reserved	ADCSOAO	Reserved	Reserved
G14	Reserved	Reserved	Reserved	EXTSYNCOUT
G15	Reserved	ADCSOCBO	Reserved	Reserved
G16	Reserved	Reserved	Reserved	Reserved
G17	Reserved	Reserved	Reserved	Reserved
G18	Reserved	Reserved	Reserved	Reserved
G19	Reserved	Reserved	Reserved	ERRORSTS
G20	Reserved	Reserved	Reserved	Reserved
G21	Reserved	Reserved	Reserved	Reserved
G22	Reserved	Reserved	Reserved	Reserved
G23	Reserved	Reserved	Reserved	Reserved
G24	Reserved	Reserved	Reserved	Reserved
G25	Reserved	Reserved	Reserved	Reserved
G26	Reserved	Reserved	Reserved	Reserved
G27	Reserved	Reserved	Reserved	Reserved
G28	Reserved	Reserved	Reserved	Reserved
G29	Reserved	Reserved	Reserved	Reserved
G30	Reserved	Reserved	Reserved	EPG1OUT0
G31	Reserved	Reserved	ERRORSTS	EPG1OUT1

### 9.2.3 X-BAR Flags

With the exception of the CMPSS signals, the ePWM X-BAR and the Output X-BAR have all of the same input signals. Due to the inputs being similar between the ePWM X-BAR and Output X-BAR, all X-BAR modules leverage a single set of input flags to indicate which input signals have been triggered. This allows software to check the input flags when an event occurs. See Figure 9-4 for more information. There is a bit allocated for each input signal in one of the XBARFLGx registers. The flag remains set until cleared through the appropriate XBARCLR register.

**Note**

Not all input sources are routed to all X-BAR modules. Refer to the X-BAR specific configuration tables for exact connections.

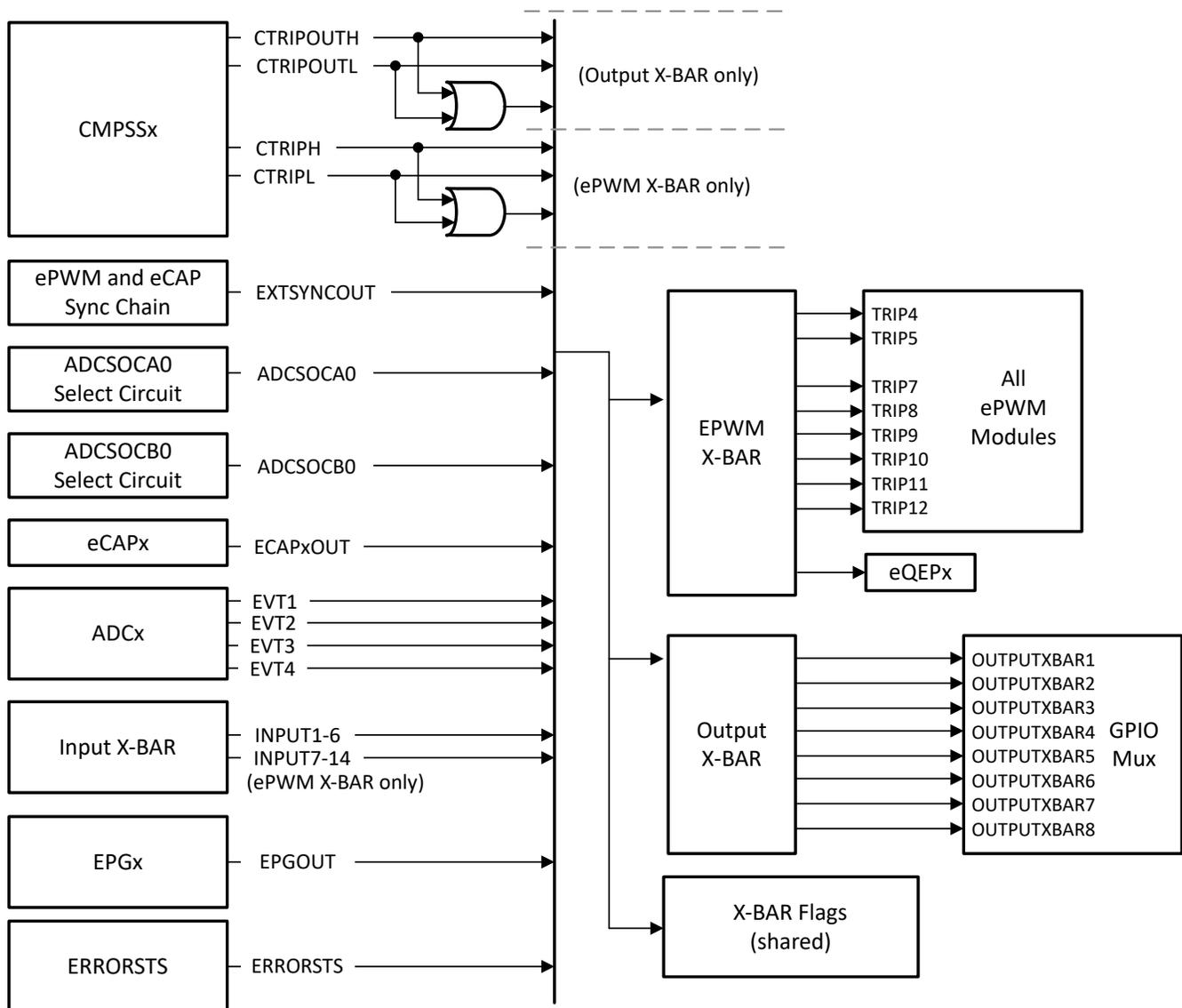


Figure 9-4. X-BAR Input Sources

## 9.3 XBAR Registers

This section describes the Crossbar registers.

### 9.3.1 XBAR Base Address Table

**Table 9-4. XBAR Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
InputXbarRegs	INPUT_XBAR_REGS	INPUTXBAR_BASE	0x0000_7900	YES
XbarRegs	XBAR_REGS	XBAR_BASE	0x0000_7920	YES
EPwmXbarRegs	EPWM_XBAR_REGS	EPWMXBAR_BASE	0x0000_7A00	YES
OutputXbarRegs	OUTPUT_XBAR_REGS	OUTPUTXBAR_BASE	0x0000_7A80	YES

### 9.3.2 INPUT\_XBAR\_REGS Registers

Table 9-5 lists the memory-mapped registers for the INPUT\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-5 should be considered as reserved locations and the register contents should not be modified.

**Table 9-5. INPUT\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	INPUT1SELECT	INPUT1 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
1h	INPUT2SELECT	INPUT2 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
2h	INPUT3SELECT	INPUT3 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
3h	INPUT4SELECT	INPUT4 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
4h	INPUT5SELECT	INPUT5 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
5h	INPUT6SELECT	INPUT6 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
6h	INPUT7SELECT	INPUT7 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
7h	INPUT8SELECT	INPUT8 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
8h	INPUT9SELECT	INPUT9 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
9h	INPUT10SELECT	INPUT10 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Ah	INPUT11SELECT	INPUT11 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Bh	INPUT12SELECT	INPUT12 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Ch	INPUT13SELECT	INPUT13 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Dh	INPUT14SELECT	INPUT14 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Eh	INPUT15SELECT	INPUT15 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
Fh	INPUT16SELECT	INPUT16 Input Select Register (GPIO0 to x)	EALLOW	<a href="#">Go</a>
1Eh	INPUTSELECTLOCK	Input Select Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-6 shows the codes that are used for access types in this section.

**Table 9-6. INPUT\_XBAR\_REGS Access Type Codes**

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

### 9.3.2.1 INPUT1SELECT Register (Offset = 0h) [Reset = FFFEh]

INPUT1SELECT is shown in [Figure 9-5](#) and described in [Table 9-7](#).

Return to the [Summary Table](#).

INPUT1 Input Select Register (GPIO0 to x)

**Figure 9-5. INPUT1SELECT Register**

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

**Table 9-7. INPUT1SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT1 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.2 INPUT2SELECT Register (Offset = 1h) [Reset = FFFEh]

INPUT2SELECT is shown in [Figure 9-6](#) and described in [Table 9-8](#).

Return to the [Summary Table](#).

INPUT2 Input Select Register (GPIO0 to x)

**Figure 9-6. INPUT2SELECT Register**

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

**Table 9-8. INPUT2SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT2 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.3 INPUT3SELECT Register (Offset = 2h) [Reset = FFFEh]

INPUT3SELECT is shown in [Figure 9-7](#) and described in [Table 9-9](#).

Return to the [Summary Table](#).

INPUT3 Input Select Register (GPIO0 to x)

**Figure 9-7. INPUT3SELECT Register**

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

**Table 9-9. INPUT3SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT3 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.4 INPUT4SELECT Register (Offset = 3h) [Reset = FFFEh]

INPUT4SELECT is shown in [Figure 9-8](#) and described in [Table 9-10](#).

Return to the [Summary Table](#).

INPUT4 Input Select Register (GPIO0 to x)

**Figure 9-8. INPUT4SELECT Register**

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

**Table 9-10. INPUT4SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT4 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.5 INPUT5SELECT Register (Offset = 4h) [Reset = FFFEh]

INPUT5SELECT is shown in [Figure 9-9](#) and described in [Table 9-11](#).

Return to the [Summary Table](#).

INPUT5 Input Select Register (GPIO0 to x)

**Figure 9-9. INPUT5SELECT Register**

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

**Table 9-11. INPUT5SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT5 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.6 INPUT6SELECT Register (Offset = 5h) [Reset = FFFEh]

INPUT6SELECT is shown in [Figure 9-10](#) and described in [Table 9-12](#).

Return to the [Summary Table](#).

INPUT6 Input Select Register (GPIO0 to x)

**Figure 9-10. INPUT6SELECT Register**

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

**Table 9-12. INPUT6SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT6 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.7 INPUT7SELECT Register (Offset = 6h) [Reset = FFFEh]

INPUT7SELECT is shown in [Figure 9-11](#) and described in [Table 9-13](#).

Return to the [Summary Table](#).

INPUT7 Input Select Register (GPIO0 to x)

**Figure 9-11. INPUT7SELECT Register**

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

**Table 9-13. INPUT7SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT7 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.8 INPUT8SELECT Register (Offset = 7h) [Reset = FFFEh]

INPUT8SELECT is shown in [Figure 9-12](#) and described in [Table 9-14](#).

Return to the [Summary Table](#).

INPUT8 Input Select Register (GPIO0 to x)

**Figure 9-12. INPUT8SELECT Register**

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

**Table 9-14. INPUT8SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT8 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.9 INPUT9SELECT Register (Offset = 8h) [Reset = FFFEh]

INPUT9SELECT is shown in [Figure 9-13](#) and described in [Table 9-15](#).

Return to the [Summary Table](#).

INPUT9 Input Select Register (GPIO0 to x)

**Figure 9-13. INPUT9SELECT Register**

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

**Table 9-15. INPUT9SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT9 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.10 INPUT10SELECT Register (Offset = 9h) [Reset = FFFEh]

INPUT10SELECT is shown in [Figure 9-14](#) and described in [Table 9-16](#).

Return to the [Summary Table](#).

INPUT10 Input Select Register (GPIO0 to x)

**Figure 9-14. INPUT10SELECT Register**

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

**Table 9-16. INPUT10SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT10 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.11 INPUT11SELECT Register (Offset = Ah) [Reset = FFFEh]

INPUT11SELECT is shown in [Figure 9-15](#) and described in [Table 9-17](#).

Return to the [Summary Table](#).

INPUT11 Input Select Register (GPIO0 to x)

**Figure 9-15. INPUT11SELECT Register**

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

**Table 9-17. INPUT11SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT11 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.12 INPUT12SELECT Register (Offset = Bh) [Reset = FFFEh]

INPUT12SELECT is shown in [Figure 9-16](#) and described in [Table 9-18](#).

Return to the [Summary Table](#).

INPUT12 Input Select Register (GPIO0 to x)

**Figure 9-16. INPUT12SELECT Register**

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

**Table 9-18. INPUT12SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT12 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.13 INPUT13SELECT Register (Offset = Ch) [Reset = FFFEh]

INPUT13SELECT is shown in [Figure 9-17](#) and described in [Table 9-19](#).

Return to the [Summary Table](#).

INPUT13 Input Select Register (GPIO0 to x)

**Figure 9-17. INPUT13SELECT Register**

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

**Table 9-19. INPUT13SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT13 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.14 INPUT14SELECT Register (Offset = Dh) [Reset = FFFEh]

INPUT14SELECT is shown in [Figure 9-18](#) and described in [Table 9-20](#).

Return to the [Summary Table](#).

INPUT14 Input Select Register (GPIO0 to x)

**Figure 9-18. INPUT14SELECT Register**

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

**Table 9-20. INPUT14SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT14 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.15 INPUT15SELECT Register (Offset = Eh) [Reset = FFFEh]

INPUT15SELECT is shown in [Figure 9-19](#) and described in [Table 9-21](#).

Return to the [Summary Table](#).

INPUT15 Input Select Register (GPIO0 to x)

**Figure 9-19. INPUT15SELECT Register**

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

**Table 9-21. INPUT15SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT15 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFFD: '1' will be driven to the destination 0xFFFFE: '1' will be driven to the destination 0xFFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.16 INPUT16SELECT Register (Offset = Fh) [Reset = FFFEh]

INPUT16SELECT is shown in [Figure 9-20](#) and described in [Table 9-22](#).

Return to the [Summary Table](#).

INPUT16 Input Select Register (GPIO0 to x)

**Figure 9-20. INPUT16SELECT Register**

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

**Table 9-22. INPUT16SELECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SELECT	R/W	FFFEh	Select GPIO for INPUT16 signal: 0x0 : Select GPIO0 0x1 : Select GPIO1 0x2 : Select GPIO2 ... 0xFFFD: '1' will be driven to the destination 0xFFFE: '1' will be driven to the destination 0xFFFF: '0' will be driven to the destination NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'. Reset type: CPU1.SYSRSn

### 9.3.2.17 INPUTSELECTLOCK Register (Offset = 1Eh) [Reset = 0000000h]

INPUTSELECTLOCK is shown in [Figure 9-21](#) and described in [Table 9-23](#).

Return to the [Summary Table](#).

Input Select Lock Register.

Any bit in this register, once set can only be cleared through SYSRSn. Write of 0 to any bit of this register has no effect. Reads to the registers which have LOCK protection are always allowed.

**Figure 9-21. 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/WOnce-0h	R/WOnce-0h						
7	6	5	4	3	2	1	0
INPUT8SELEC T	INPUT7SELEC T	INPUT6SELEC T	INPUT5SELEC T	INPUT4SELEC T	INPUT3SELEC T	INPUT2SELEC T	INPUT1SELEC T
R/WOnce-0h	R/WOnce-0h						

**Table 9-23. INPUTSELECTLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15	INPUT16SELECT	R/WOnce	0h	Lock bit for INPUT16SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
14	INPUT15SELECT	R/WOnce	0h	Lock bit for INPUT15SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
13	INPUT14SELECT	R/WOnce	0h	Lock bit for INPUT14SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
12	INPUT13SELECT	R/WOnce	0h	Lock bit for INPUT13SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
11	INPUT12SELECT	R/WOnce	0h	Lock bit for INPUT12SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
10	INPUT11SELECT	R/WOnce	0h	Lock bit for INPUT11SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn

**Table 9-23. INPUTSELECTLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	INPUT10SELECT	R/WOnce	0h	Lock bit for INPUT10SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
8	INPUT9SELECT	R/WOnce	0h	Lock bit for INPUT9SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
7	INPUT8SELECT	R/WOnce	0h	Lock bit for INPUT8SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
6	INPUT7SELECT	R/WOnce	0h	Lock bit for INPUT7SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
5	INPUT6SELECT	R/WOnce	0h	Lock bit for INPUT6SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
4	INPUT5SELECT	R/WOnce	0h	Lock bit for INPUT5SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
3	INPUT4SELECT	R/WOnce	0h	Lock bit for INPUT4SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
2	INPUT3SELECT	R/WOnce	0h	Lock bit for INPUT3SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
1	INPUT2SELECT	R/WOnce	0h	Lock bit for INPUT2SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn
0	INPUT1SELECT	R/WOnce	0h	Lock bit for INPUT1SELECT Register 0: Register is not locked 1: Register is locked Reset type: CPU1.SYSRSn

### 9.3.3 XBAR\_REGS Registers

Table 9-24 lists the memory-mapped registers for the XBAR\_REGS registers. All register offset addresses not listed in Table 9-24 should be considered as reserved locations and the register contents should not be modified.

**Table 9-24. XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	XBARFLG1	X-Bar Input Flag Register 1		<a href="#">Go</a>
2h	XBARFLG2	X-Bar Input Flag Register 2		<a href="#">Go</a>
4h	XBARFLG3	X-Bar Input Flag Register 3		<a href="#">Go</a>
6h	XBARFLG4	X-Bar Input Flag Register 4		<a href="#">Go</a>
8h	XBARCLR1	X-Bar Input Flag Clear Register 1		<a href="#">Go</a>
Ah	XBARCLR2	X-Bar Input Flag Clear Register 2		<a href="#">Go</a>
Ch	XBARCLR3	X-Bar Input Flag Clear Register 3		<a href="#">Go</a>
Eh	XBARCLR4	X-Bar Input Flag Clear Register 4		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-25 shows the codes that are used for access types in this section.

**Table 9-25. XBAR\_REGS Access Type Codes**

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

### 9.3.3.1 XBARFLG1 Register (Offset = 0h) [Reset = 0000000h]

XBARFLG1 is shown in [Figure 9-22](#) and described in [Table 9-26](#).

Return to the [Summary Table](#).

X-Bar Input Flag Register 1

**Figure 9-22. XBARFLG1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-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-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-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-0h							

**Table 9-26. XBARFLG1 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	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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
18	CMPSS2_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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
16	CMPSS1_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 Reset type: CPU1.SYSRSn
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	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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn

### 9.3.3.2 XBARFLG2 Register (Offset = 2h) [Reset = 0000000h]

XBARFLG2 is shown in [Figure 9-23](#) and described in [Table 9-27](#).

Return to the [Summary Table](#).

X-Bar Input Flag Register 2

**Figure 9-23. XBARFLG2 Register**

31	30	29	28	27	26	25	24
ADCCEVT1	RESERVED	RESERVED	RESERVED	RESERVED	ADCAEVT4	ADCAEVT3	ADCAEVT2
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
ADCAEVT1	EXTSYNCOUT	RESERVED	RESERVED	RESERVED	RESERVED	ECAP2_OUT	ECAP1_OUT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
INPUT14	INPUT13	INPUT12	INPUT11	INPUT10	INPUT9	INPUT8	INPUT7
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
ADCSOCB	ADCSOCA	INPUT6	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-27. 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 Reset type: CPU1.SYSRSn
30	RESERVED	R	0h	Reserved
29	RESERVED	R	0h	Reserved
28	RESERVED	R	0h	Reserved
27	RESERVED	R	0h	Reserved
26	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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
21	RESERVED	R	0h	Reserved
20	RESERVED	R	0h	Reserved
19	RESERVED	R	0h	Reserved
18	RESERVED	R	0h	Reserved
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
15	INPUT14	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 Reset type: CPU1.SYSRSn
14	INPUT13	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 Reset type: CPU1.SYSRSn
13	INPUT12	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 Reset type: CPU1.SYSRSn

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	INPUT11	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 Reset type: CPU1.SYSRSn
11	INPUT10	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 Reset type: CPU1.SYSRSn
10	INPUT9	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 Reset type: CPU1.SYSRSn
9	INPUT8	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 Reset type: CPU1.SYSRSn
8	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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
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 Reset type: CPU1.SYSRSn
5	INPUT6	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 Reset type: CPU1.SYSRSn

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: CPU1.SYSRSn
3	INPUT4	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 Reset type: CPU1.SYSRSn
2	INPUT3	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 Reset type: CPU1.SYSRSn
1	INPUT2	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 Reset type: CPU1.SYSRSn
0	INPUT1	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 Reset type: CPU1.SYSRSn

### 9.3.3.3 XBARFLG3 Register (Offset = 4h) [Reset = 0000000h]

XBARFLG3 is shown in [Figure 9-24](#) and described in [Table 9-28](#).

Return to the [Summary Table](#).

X-Bar Input Flag Register 3

**Figure 9-24. XBARFLG3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	ADCCEVT4	ADCCEVT3	ADCCEVT2
R-0h							

**Table 9-28. XBARFLG3 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	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved

**Table 9-28. XBARFLG3 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	ADCCEVT4	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 Reset type: CPU1.SYSRSn
1	ADCCEVT3	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 Reset type: CPU1.SYSRSn
0	ADCCEVT2	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 Reset type: CPU1.SYSRSn

### 9.3.3.4 XBARFLG4 Register (Offset = 6h) [Reset = 0000000h]

XBARFLG4 is shown in [Figure 9-25](#) and described in [Table 9-29](#).

Return to the [Summary Table](#).

X-Bar Input Flag Register 4

**Figure 9-25. XBARFLG4 Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	ERRORSTS_ERROR	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	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-29. XBARFLG4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	RESERVED	R	0h	Reserved
29	RESERVED	R	0h	Reserved
28	ERRORSTS_ERROR	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: ERRORSTS_ERROR input was triggered 0: ERRORSTS_ERROR Input was not triggered Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
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 9-29. XBARFLG4 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	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved
5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	RESERVED	R	0h	Reserved
1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

### 9.3.3.5 XBARCLR1 Register (Offset = 8h) [Reset = 0000000h]

XBARCLR1 is shown in [Figure 9-26](#) and described in [Table 9-30](#).

Return to the [Summary Table](#).

X-Bar Input Flag Clear Register 1

**Figure 9-26. XBARCLR1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W1S-0h							
23	22	21	20	19	18	17	16
CMPSS4_CTRI POUTH	CMPSS4_CTRI POUTL	CMPSS3_CTRI POUTH	CMPSS3_CTRI POUTL	CMPSS2_CTRI POUTH	CMPSS2_CTRI POUTL	CMPSS1_CTRI POUTH	CMPSS1_CTRI POUTL
R-0/W1S-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0/W1S-0h							
7	6	5	4	3	2	1	0
CMPSS4_CTRI PH	CMPSS4_CTRI PL	CMPSS3_CTRI PH	CMPSS3_CTRI PL	CMPSS2_CTRI PH	CMPSS2_CTRI PL	CMPSS1_CTRI PH	CMPSS1_CTRI PL
R-0/W1S-0h							

**Table 9-30. XBARCLR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W1S	0h	Reserved
30	RESERVED	R-0/W1S	0h	Reserved
29	RESERVED	R-0/W1S	0h	Reserved
28	RESERVED	R-0/W1S	0h	Reserved
27	RESERVED	R-0/W1S	0h	Reserved
26	RESERVED	R-0/W1S	0h	Reserved
25	RESERVED	R-0/W1S	0h	Reserved
24	RESERVED	R-0/W1S	0h	Reserved
23	CMPSS4_CTRIPOUTH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	CMPSS4_CTRIPOUTL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	CMPSS3_CTRIPOUTH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
20	CMPSS3_CTRIPOUTL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
19	CMPSS2_CTRIPOUTH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-30. XBARCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	CMPSS2_CTRIPOUTL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
17	CMPSS1_CTRIPOUTH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	CMPSS1_CTRIPOUTL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	RESERVED	R-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	CMPSS4_CTRIPH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	CMPSS4_CTRIPL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
5	CMPSS3_CTRIPH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
4	CMPSS3_CTRIPL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
3	CMPSS2_CTRIPH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
2	CMPSS2_CTRIPL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	CMPSS1_CTRIPH	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	CMPSS1_CTRIPL	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARFLG1 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.3.6 XBARCLR2 Register (Offset = Ah) [Reset = 0000000h]

XBARCLR2 is shown in [Figure 9-27](#) and described in [Table 9-31](#).

Return to the [Summary Table](#).

X-Bar Input Flag Clear Register 2

**Figure 9-27. XBARCLR2 Register**

31	30	29	28	27	26	25	24
ADCCEVT1	RESERVED	RESERVED	RESERVED	RESERVED	ADCAEVT4	ADCAEVT3	ADCAEVT2
R-0/W1S-0h							
23	22	21	20	19	18	17	16
ADCAEVT1	EXTSYNCOUT	RESERVED	RESERVED	RESERVED	RESERVED	ECAP2_OUT	ECAP1_OUT
R-0/W1S-0h							
15	14	13	12	11	10	9	8
INPUT14	INPUT13	INPUT12	INPUT11	INPUT10	INPUT9	INPUT8	INPUT7
R-0/W1S-0h							
7	6	5	4	3	2	1	0
ADCSOCB	ADCSOCA	INPUT6	INPUT5	INPUT4	INPUT3	INPUT2	INPUT1
R-0/W1S-0h							

**Table 9-31. XBARCLR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	ADCCEVT1	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
30	RESERVED	R-0/W1S	0h	Reserved
29	RESERVED	R-0/W1S	0h	Reserved
28	RESERVED	R-0/W1S	0h	Reserved
27	RESERVED	R-0/W1S	0h	Reserved
26	ADCAEVT4	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
25	ADCAEVT3	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
24	ADCAEVT2	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
23	ADCAEVT1	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
22	EXTSYNCOUT	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
21	RESERVED	R-0/W1S	0h	Reserved
20	RESERVED	R-0/W1S	0h	Reserved

**Table 9-31. XBARCLR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	RESERVED	R-0/W1S	0h	Reserved
18	RESERVED	R-0/W1S	0h	Reserved
17	ECAP2_OUT	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
16	ECAP1_OUT	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
15	INPUT14	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
14	INPUT13	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
13	INPUT12	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
12	INPUT11	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
11	INPUT10	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
10	INPUT9	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
9	INPUT8	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
8	INPUT7	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
7	ADCSOCB	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
6	ADCSOCA	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
5	INPUT6	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

**Table 9-31. XBARCLR2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	INPUT5	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
3	INPUT4	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
2	INPUT3	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	INPUT2	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	INPUT1	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT2 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.3.7 XBARCLR3 Register (Offset = Ch) [Reset = 0000000h]

XBARCLR3 is shown in [Figure 9-28](#) and described in [Table 9-32](#).

Return to the [Summary Table](#).

X-Bar Input Flag Clear Register 3

**Figure 9-28. XBARCLR3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0/W1S-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0/W1S-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0/W1S-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	ADCCEVT4	ADCCEVT3	ADCCEVT2
R-0/W1S-0h							

**Table 9-32. XBARCLR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R-0/W1S	0h	Reserved
30	RESERVED	R-0/W1S	0h	Reserved
29	RESERVED	R-0/W1S	0h	Reserved
28	RESERVED	R-0/W1S	0h	Reserved
27	RESERVED	R-0/W1S	0h	Reserved
26	RESERVED	R-0/W1S	0h	Reserved
25	RESERVED	R-0/W1S	0h	Reserved
24	RESERVED	R-0/W1S	0h	Reserved
23	RESERVED	R-0/W1S	0h	Reserved
22	RESERVED	R-0/W1S	0h	Reserved
21	RESERVED	R-0/W1S	0h	Reserved
20	RESERVED	R-0/W1S	0h	Reserved
19	RESERVED	R-0/W1S	0h	Reserved
18	RESERVED	R-0/W1S	0h	Reserved
17	RESERVED	R-0/W1S	0h	Reserved
16	RESERVED	R-0/W1S	0h	Reserved
15	RESERVED	R-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	RESERVED	R-0/W1S	0h	Reserved

**Table 9-32. XBARCLR3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	RESERVED	R-0/W1S	0h	Reserved
5	RESERVED	R-0/W1S	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	ADCCEVT4	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
1	ADCCEVT3	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
0	ADCCEVT2	R-0/W1S	0h	Writing 1 to a bit in this register clears the corresponding bit in the XBARILAT3 register. Writing 0 has no effect Reset type: CPU1.SYSRSn

### 9.3.3.8 XBARCLR4 Register (Offset = Eh) [Reset = 0000000h]

XBARCLR4 is shown in [Figure 9-29](#) and described in [Table 9-33](#).

Return to the [Summary Table](#).

X-Bar Input Flag Clear Register 4

**Figure 9-29. XBARCLR4 Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	ERRORSTS_ERROR	RESERVED	RESERVED	RESERVED	RESERVED
R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
23	22	21	20	19	18	17	16
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	RESERVED
R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-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

**Table 9-33. XBARCLR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30	RESERVED	R-0/W1S	0h	Reserved
29	RESERVED	R-0/W1S	0h	Reserved
28	ERRORSTS_ERROR	R-0/W1S	0h	Writing 1 to this bit clears the ERRORSTS_ERROR bit in the XBARCLR4 register. Writing 0 has no effect Reset type: CPU1.SYSRSn
27	RESERVED	R-0/W1S	0h	Reserved
26	RESERVED	R-0/W1S	0h	Reserved
25	RESERVED	R-0/W1S	0h	Reserved
24	RESERVED	R-0/W1S	0h	Reserved
23	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-0/W1S	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13	RESERVED	R-0/W1S	0h	Reserved
12	RESERVED	R-0/W1S	0h	Reserved
11	RESERVED	R-0/W1S	0h	Reserved
10	RESERVED	R-0/W1S	0h	Reserved

**Table 9-33. XBARCLR4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R-0/W1S	0h	Reserved
8	RESERVED	R-0/W1S	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

### 9.3.4 EPWM\_XBAR\_REGS Registers

Table 9-34 lists the memory-mapped registers for the EPWM\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-34 should be considered as reserved locations and the register contents should not be modified.

**Table 9-34. EPWM\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TRIP4MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	<a href="#">Go</a>
2h	TRIP4MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP4	EALLOW	<a href="#">Go</a>
4h	TRIP5MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	<a href="#">Go</a>
6h	TRIP5MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP5	EALLOW	<a href="#">Go</a>
8h	TRIP7MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	<a href="#">Go</a>
Ah	TRIP7MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP7	EALLOW	<a href="#">Go</a>
Ch	TRIP8MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	<a href="#">Go</a>
Eh	TRIP8MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP8	EALLOW	<a href="#">Go</a>
10h	TRIP9MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	<a href="#">Go</a>
12h	TRIP9MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP9	EALLOW	<a href="#">Go</a>
14h	TRIP10MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	<a href="#">Go</a>
16h	TRIP10MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP10	EALLOW	<a href="#">Go</a>
18h	TRIP11MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	<a href="#">Go</a>
1Ah	TRIP11MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP11	EALLOW	<a href="#">Go</a>
1Ch	TRIP12MUX0TO15CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	<a href="#">Go</a>
1Eh	TRIP12MUX16TO31CFG	ePWM XBAR Mux Configuration for TRIP12	EALLOW	<a href="#">Go</a>
20h	TRIP4MUXENABLE	ePWM XBAR Mux Enable for TRIP4	EALLOW	<a href="#">Go</a>
22h	TRIP5MUXENABLE	ePWM XBAR Mux Enable for TRIP5	EALLOW	<a href="#">Go</a>
24h	TRIP7MUXENABLE	ePWM XBAR Mux Enable for TRIP7	EALLOW	<a href="#">Go</a>
26h	TRIP8MUXENABLE	ePWM XBAR Mux Enable for TRIP8	EALLOW	<a href="#">Go</a>
28h	TRIP9MUXENABLE	ePWM XBAR Mux Enable for TRIP9	EALLOW	<a href="#">Go</a>
2Ah	TRIP10MUXENABLE	ePWM XBAR Mux Enable for TRIP10	EALLOW	<a href="#">Go</a>
2Ch	TRIP11MUXENABLE	ePWM XBAR Mux Enable for TRIP11	EALLOW	<a href="#">Go</a>
2Eh	TRIP12MUXENABLE	ePWM XBAR Mux Enable for TRIP12	EALLOW	<a href="#">Go</a>
38h	TRIPOUTINV	ePWM XBAR Output Inversion Register	EALLOW	<a href="#">Go</a>
3Eh	TRIPLOCK	ePWM XBAR Configuration Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-35 shows the codes that are used for access types in this section.

**Table 9-35. EPWM\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
WOnce	W Once	Write Set once
Reset or Default Value		
-n		Value after reset or the default value

**Table 9-35. EPWM\_XBAR\_REGS Access Type Codes (continued)**

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

### 9.3.4.1 TRIP4MUX0TO15CFG Register (Offset = 0h) [Reset = 0000000h]

TRIP4MUX0TO15CFG is shown in [Figure 9-30](#) and described in [Table 9-36](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-30. TRIP4MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.2 TRIP4MUX16TO31CFG Register (Offset = 2h) [Reset = 0000000h]

TRIP4MUX16TO31CFG is shown in [Figure 9-31](#) and described in [Table 9-37](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-31. TRIP4MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP4 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.3 TRIP5MUX0TO15CFG Register (Offset = 4h) [Reset = 0000000h]

TRIP5MUX0TO15CFG is shown in [Figure 9-32](#) and described in [Table 9-38](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-32. TRIP5MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.4 TRIP5MUX16TO31CFG Register (Offset = 6h) [Reset = 0000000h]

TRIP5MUX16TO31CFG is shown in [Figure 9-33](#) and described in [Table 9-39](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-33. TRIP5MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP5 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.5 TRIP7MUX0TO15CFG Register (Offset = 8h) [Reset = 0000000h]

TRIP7MUX0TO15CFG is shown in [Figure 9-34](#) and described in [Table 9-40](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-34. TRIP7MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.6 TRIP7MUX16TO31CFG Register (Offset = Ah) [Reset = 0000000h]

TRIP7MUX16TO31CFG is shown in [Figure 9-35](#) and described in [Table 9-41](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-35. TRIP7MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP7 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.7 TRIP8MUX0TO15CFG Register (Offset = Ch) [Reset = 0000000h]

TRIP8MUX0TO15CFG is shown in [Figure 9-36](#) and described in [Table 9-42](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-36. TRIP8MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.8 TRIP8MUX16TO31CFG Register (Offset = Eh) [Reset = 0000000h]

TRIP8MUX16TO31CFG is shown in [Figure 9-37](#) and described in [Table 9-43](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-37. TRIP8MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP8 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.9 TRIP9MUX0TO15CFG Register (Offset = 10h) [Reset = 0000000h]

TRIP9MUX0TO15CFG is shown in [Figure 9-38](#) and described in [Table 9-44](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-38. TRIP9MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.10 TRIP9MUX16TO31CFG Register (Offset = 12h) [Reset = 0000000h]

TRIP9MUX16TO31CFG is shown in [Figure 9-39](#) and described in [Table 9-45](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-39. TRIP9MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP9 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.11 TRIP10MUX0TO15CFG Register (Offset = 14h) [Reset = 0000000h]

TRIP10MUX0TO15CFG is shown in [Figure 9-40](#) and described in [Table 9-46](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-40. TRIP10MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.12 TRIP10MUX16TO31CFG Register (Offset = 16h) [Reset = 0000000h]

TRIP10MUX16TO31CFG is shown in [Figure 9-41](#) and described in [Table 9-47](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-41. TRIP10MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP10 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.13 TRIP11MUX0TO15CFG Register (Offset = 18h) [Reset = 0000000h]

TRIP11MUX0TO15CFG is shown in [Figure 9-42](#) and described in [Table 9-48](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-42. TRIP11MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.14 TRIP11MUX16TO31CFG Register (Offset = 1Ah) [Reset = 0000000h]

TRIP11MUX16TO31CFG is shown in [Figure 9-43](#) and described in [Table 9-49](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-43. TRIP11MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP11 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.15 TRIP12MUX0TO15CFG Register (Offset = 1Ch) [Reset = 0000000h]

TRIP12MUX0TO15CFG is shown in [Figure 9-44](#) and described in [Table 9-50](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-44. TRIP12MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.16 TRIP12MUX16TO31CFG Register (Offset = 1Eh) [Reset = 0000000h]

TRIP12MUX16TO31CFG is shown in [Figure 9-45](#) and described in [Table 9-51](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-45. TRIP12MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for EPWM-XBAR TRIP12 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.17 TRIP4MUXENABLE Register (Offset = 20h) [Reset = 0000000h]

TRIP4MUXENABLE is shown in [Figure 9-46](#) and described in [Table 9-52](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP4

**Figure 9-46. TRIP4MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux31 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux30 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux29 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux28 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux26 is disabled 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. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux25 is disabled 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. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux24 is disabled 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP4 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP4 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP4 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

### 9.3.4.18 TRIP5MUXENABLE Register (Offset = 22h) [Reset = 0000000h]

TRIP5MUXENABLE is shown in [Figure 9-47](#) and described in [Table 9-53](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP5

**Figure 9-47. TRIP5MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-53. 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 disabled 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. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux30 is disabled 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. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux29 is disabled 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. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux28 is disabled 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. Reset type: CPU1.SYSRSn

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux26 is disabled 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. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux25 is disabled 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. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux24 is disabled 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP5 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP5 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP5 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.19 TRIP7MUXENABLE Register (Offset = 24h) [Reset = 0000000h]

TRIP7MUXENABLE is shown in [Figure 9-48](#) and described in [Table 9-54](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP7

**Figure 9-48. TRIP7MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-54. 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 disabled 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. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux30 is disabled 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. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux29 is disabled 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. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux28 is disabled 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. Reset type: CPU1.SYSRSn

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux26 is disabled 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. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux25 is disabled 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. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux24 is disabled 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP7 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP7 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP7 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.20 TRIP8MUXENABLE Register (Offset = 26h) [Reset = 0000000h]

TRIP8MUXENABLE is shown in [Figure 9-49](#) and described in [Table 9-55](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP8

**Figure 9-49. TRIP8MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-55. 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 disabled 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. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux30 is disabled 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. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux29 is disabled 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. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux28 is disabled 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. Reset type: CPU1.SYSRSn

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux26 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux25 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux24 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP8 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP8 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP8 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.21 TRIP9MUXENABLE Register (Offset = 28h) [Reset = 0000000h]

TRIP9MUXENABLE is shown in [Figure 9-50](#) and described in [Table 9-56](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP9

**Figure 9-50. TRIP9MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux31 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux30 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux29 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux28 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux26 is disabled 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. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux25 is disabled 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. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux24 is disabled 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP9 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP9 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP9 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.22 TRIP10MUXENABLE Register (Offset = 2Ah) [Reset = 0000000h]

TRIP10MUXENABLE is shown in [Figure 9-51](#) and described in [Table 9-57](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP10

**Figure 9-51. TRIP10MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux31 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux30 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux29 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux28 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux26 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux25 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux24 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP10 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP10 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP10 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

### 9.3.4.23 TRIP11MUXENABLE Register (Offset = 2Ch) [Reset = 0000000h]

TRIP11MUXENABLE is shown in [Figure 9-52](#) and described in [Table 9-58](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP11

**Figure 9-52. TRIP11MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux31 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux31 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux30 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux30 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux29 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux29 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux28 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux28 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux26 is disabled 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. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux25 is disabled 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. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux24 is disabled 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP11 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP11 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP11 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.24 TRIP12MUXENABLE Register (Offset = 2Eh) [Reset = 0000000h]

TRIP12MUXENABLE is shown in [Figure 9-53](#) and described in [Table 9-59](#).

Return to the [Summary Table](#).

ePWM XBAR Mux Enable for TRIP12

**Figure 9-53. TRIP12MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-59. 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 disabled 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. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux30 is disabled 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. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux29 is disabled 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. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux28 is disabled 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. Reset type: CPU1.SYSRSn

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27	MUX27	R/W	0h	Selects the output of Mux27 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux27 is disabled 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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux26 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux26 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux25 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux25 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux24 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux24 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux23 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux23 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux22 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux22 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux21 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux21 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux20 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux20 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux19 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux19 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux18 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux18 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux17 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux17 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux16 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux16 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux15 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux15 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux14 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux14 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux13 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux13 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux12 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux12 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux11 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux11 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux10 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux10 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux9 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux9 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux8 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux8 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux7 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux7 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux6 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux6 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux5 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux5 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux4 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux4 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux3 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux3 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux2 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux2 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux1 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux1 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive TRIP12 of EPWM-XBAR 0: Respective output of Mux0 is disabled to drive the TRIP12 of EPWM-XBAR 1: Respective output of Mux0 is enabled to drive the TRIP12 of EPWM-XBAR Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.25 TRIPOUTINV Register (Offset = 38h) [Reset = 0000000h]

TRIP0UTINV is shown in [Figure 9-54](#) and described in [Table 9-60](#).

Return to the [Summary Table](#).

ePWM XBAR Output Inversion Register

**Figure 9-54. TRIPOUTINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
TRIP12	TRIP11	TRIP10	TRIP9	TRIP8	TRIP7	TRIP5	TRIP4
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 9-60. TRIPOUTINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	TRIP12	R/W	0h	Selects polarity for TRIP12 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	TRIP11	R/W	0h	Selects polarity for TRIP11 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	TRIP10	R/W	0h	Selects polarity for TRIP10 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	TRIP9	R/W	0h	Selects polarity for TRIP9 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3	TRIP8	R/W	0h	Selects polarity for TRIP8 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	TRIP7	R/W	0h	Selects polarity for TRIP7 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-60. 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. Reset type: CPU1.SYSRSn
0	TRIP4	R/W	0h	Selects polarity for TRIP4 of EPWM-XBAR 0: drives active high output 1: drives active-low output Refer to the EPWM X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.4.26 TRIPLOCK Register (Offset = 3Eh) [Reset = 0000000h]

TRIPLOCK is shown in [Figure 9-55](#) and described in [Table 9-61](#).

Return to the [Summary Table](#).

ePWM XBAR Configuration Lock register

**Figure 9-55. TRIPLOCK Register**

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

**Table 9-61. TRIPLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Bit-0 of this register can be set only if KEY= 0x5a5a Reset type: CPU1.SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/WOnce	0h	Locks the configuration for EPWM-XBAR. Once the configuration is locked, writes to the below registers for EPWM-XBAR is blocked. Registers Affected by the LOCK mechanism: EPWM-XBAROUTyMUX0TO15CFG EPWM-XBAROUTyMUX16TO31CFG EPWM-XBAROUTyMUXENABLE EPWM-XBAROUTLATEN EPWM-XBAROUTINV 0: Writes to the above registers are allowed 1: Writes to the above registers are blocked Note: [1] LOCK mechanism only applies to writes. Reads are never blocked. Reset type: CPU1.SYSRSn

### 9.3.5 OUTPUT\_XBAR\_REGS Registers

Table 9-62 lists the memory-mapped registers for the OUTPUT\_XBAR\_REGS registers. All register offset addresses not listed in Table 9-62 should be considered as reserved locations and the register contents should not be modified.

**Table 9-62. OUTPUT\_XBAR\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	OUTPUT1MUX0TO15CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	<a href="#">Go</a>
2h	OUTPUT1MUX16TO31CFG	Output X-BAR Mux Configuration for Output 1	EALLOW	<a href="#">Go</a>
4h	OUTPUT2MUX0TO15CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	<a href="#">Go</a>
6h	OUTPUT2MUX16TO31CFG	Output X-BAR Mux Configuration for Output 2	EALLOW	<a href="#">Go</a>
8h	OUTPUT3MUX0TO15CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	<a href="#">Go</a>
Ah	OUTPUT3MUX16TO31CFG	Output X-BAR Mux Configuration for Output 3	EALLOW	<a href="#">Go</a>
Ch	OUTPUT4MUX0TO15CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	<a href="#">Go</a>
Eh	OUTPUT4MUX16TO31CFG	Output X-BAR Mux Configuration for Output 4	EALLOW	<a href="#">Go</a>
10h	OUTPUT5MUX0TO15CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	<a href="#">Go</a>
12h	OUTPUT5MUX16TO31CFG	Output X-BAR Mux Configuration for Output 5	EALLOW	<a href="#">Go</a>
14h	OUTPUT6MUX0TO15CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	<a href="#">Go</a>
16h	OUTPUT6MUX16TO31CFG	Output X-BAR Mux Configuration for Output 6	EALLOW	<a href="#">Go</a>
18h	OUTPUT7MUX0TO15CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	<a href="#">Go</a>
1Ah	OUTPUT7MUX16TO31CFG	Output X-BAR Mux Configuration for Output 7	EALLOW	<a href="#">Go</a>
1Ch	OUTPUT8MUX0TO15CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	<a href="#">Go</a>
1Eh	OUTPUT8MUX16TO31CFG	Output X-BAR Mux Configuration for Output 8	EALLOW	<a href="#">Go</a>
20h	OUTPUT1MUXENABLE	Output X-BAR Mux Enable for Output 1	EALLOW	<a href="#">Go</a>
22h	OUTPUT2MUXENABLE	Output X-BAR Mux Enable for Output 2	EALLOW	<a href="#">Go</a>
24h	OUTPUT3MUXENABLE	Output X-BAR Mux Enable for Output 3	EALLOW	<a href="#">Go</a>
26h	OUTPUT4MUXENABLE	Output X-BAR Mux Enable for Output 4	EALLOW	<a href="#">Go</a>
28h	OUTPUT5MUXENABLE	Output X-BAR Mux Enable for Output 5	EALLOW	<a href="#">Go</a>
2Ah	OUTPUT6MUXENABLE	Output X-BAR Mux Enable for Output 6	EALLOW	<a href="#">Go</a>
2Ch	OUTPUT7MUXENABLE	Output X-BAR Mux Enable for Output 7	EALLOW	<a href="#">Go</a>
2Eh	OUTPUT8MUXENABLE	Output X-BAR Mux Enable for Output 8	EALLOW	<a href="#">Go</a>
30h	OUTPUTLATCH	Output X-BAR Output Latch		<a href="#">Go</a>
32h	OUTPUTLATCHCLR	Output X-BAR Output Latch Clear		<a href="#">Go</a>
34h	OUTPUTLATCHFRC	Output X-BAR Output Latch Clear		<a href="#">Go</a>
36h	OUTPUTLATCHENABLE	Output X-BAR Output Latch Enable	EALLOW	<a href="#">Go</a>
38h	OUTPUTINV	Output X-BAR Output Inversion	EALLOW	<a href="#">Go</a>
3Eh	OUTPUTLOCK	Output X-BAR Configuration Lock register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 9-63 shows the codes that are used for access types in this section.

**Table 9-63. OUTPUT\_XBAR\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write

**Table 9-63. OUTPUT\_XBAR\_REGS Access Type Codes (continued)**

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

### 9.3.5.1 OUTPUT1MUX0TO15CFG Register (Offset = 0h) [Reset = 0000000h]

OUTPUT1MUX0TO15CFG is shown in [Figure 9-56](#) and described in [Table 9-64](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 1

**Figure 9-56. OUTPUT1MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT1 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT1 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT1 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT1 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT1 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT1 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT1 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT1 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT1 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT1 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT1 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT1 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT1 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT1 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT1 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT1 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.2 OUTPUT1MUX16TO31CFG Register (Offset = 2h) [Reset = 0000000h]

OUTPUT1MUX16TO31CFG is shown in [Figure 9-57](#) and described in [Table 9-65](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 1

**Figure 9-57. OUTPUT1MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT1 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT1 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT1 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT1 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT1 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT1 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT1 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT1 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT1 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT1 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT1 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT1 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT1 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT1 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT1 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT1 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.3 OUTPUT2MUX0TO15CFG Register (Offset = 4h) [Reset = 0000000h]

OUTPUT2MUX0TO15CFG is shown in [Figure 9-58](#) and described in [Table 9-66](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 2

**Figure 9-58. OUTPUT2MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT2 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT2 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT2 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT2 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT2 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT2 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT2 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT2 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT2 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT2 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT2 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT2 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT2 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT2 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT2 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT2 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.4 OUTPUT2MUX16TO31CFG Register (Offset = 6h) [Reset = 0000000h]

OUTPUT2MUX16TO31CFG is shown in [Figure 9-59](#) and described in [Table 9-67](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 2

**Figure 9-59. OUTPUT2MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT2 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT2 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT2 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT2 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT2 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT2 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT2 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT2 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT2 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT2 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT2 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT2 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT2 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT2 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT2 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT2 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.5 OUTPUT3MUX0TO15CFG Register (Offset = 8h) [Reset = 0000000h]

OUTPUT3MUX0TO15CFG is shown in [Figure 9-60](#) and described in [Table 9-68](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 3

**Figure 9-60. OUTPUT3MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT3 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT3 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT3 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT3 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT3 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT3 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT3 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT3 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT3 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT3 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT3 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT3 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT3 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT3 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT3 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT3 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.6 OUTPUT3MUX16TO31CFG Register (Offset = Ah) [Reset = 0000000h]

OUTPUT3MUX16TO31CFG is shown in [Figure 9-61](#) and described in [Table 9-69](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 3

**Figure 9-61. OUTPUT3MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT3 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT3 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT3 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT3 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT3 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT3 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT3 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT3 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT3 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT3 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT3 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT3 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT3 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT3 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT3 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT3 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.7 OUTPUT4MUX0TO15CFG Register (Offset = Ch) [Reset = 0000000h]

OUTPUT4MUX0TO15CFG is shown in [Figure 9-62](#) and described in [Table 9-70](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 4

**Figure 9-62. OUTPUT4MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT4 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT4 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT4 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT4 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT4 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT4 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT4 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT4 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT4 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT4 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT4 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT4 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT4 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT4 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT4 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT4 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.8 OUTPUT4MUX16TO31CFG Register (Offset = Eh) [Reset = 0000000h]

OUTPUT4MUX16TO31CFG is shown in [Figure 9-63](#) and described in [Table 9-71](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 4

**Figure 9-63. OUTPUT4MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT4 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT4 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT4 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT4 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT4 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT4 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT4 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT4 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT4 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT4 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT4 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT4 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT4 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT4 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT4 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT4 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.9 OUTPUT5MUX0TO15CFG Register (Offset = 10h) [Reset = 0000000h]

OUTPUT5MUX0TO15CFG is shown in [Figure 9-64](#) and described in [Table 9-72](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 5

**Figure 9-64. OUTPUT5MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT5 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT5 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT5 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT5 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT5 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT5 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT5 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT5 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT5 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT5 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT5 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT5 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT5 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT5 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT5 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT5 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.10 OUTPUT5MUX16TO31CFG Register (Offset = 12h) [Reset = 0000000h]

OUTPUT5MUX16TO31CFG is shown in [Figure 9-65](#) and described in [Table 9-73](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 5

**Figure 9-65. OUTPUT5MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT5 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT5 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT5 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT5 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT5 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT5 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT5 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT5 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT5 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT5 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT5 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT5 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT5 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT5 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT5 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT5 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.11 OUTPUT6MUX0TO15CFG Register (Offset = 14h) [Reset = 0000000h]

OUTPUT6MUX0TO15CFG is shown in [Figure 9-66](#) and described in [Table 9-74](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 6

**Figure 9-66. OUTPUT6MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT6 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT6 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT6 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT6 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT6 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT6 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT6 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT6 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT6 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT6 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT6 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT6 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT6 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT6 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT6 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT6 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.12 OUTPUT6MUX16TO31CFG Register (Offset = 16h) [Reset = 0000000h]

OUTPUT6MUX16TO31CFG is shown in [Figure 9-67](#) and described in [Table 9-75](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 6

**Figure 9-67. OUTPUT6MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT6 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT6 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT6 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT6 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT6 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT6 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT6 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT6 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT6 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT6 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT6 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT6 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT6 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT6 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT6 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT6 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.13 OUTPUT7MUX0TO15CFG Register (Offset = 18h) [Reset = 0000000h]

OUTPUT7MUX0TO15CFG is shown in [Figure 9-68](#) and described in [Table 9-76](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 7

**Figure 9-68. OUTPUT7MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT7 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT7 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT7 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT7 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT7 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT7 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT7 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT7 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT7 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT7 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT7 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT7 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT7 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT7 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT7 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT7 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.14 OUTPUT7MUX16TO31CFG Register (Offset = 1Ah) [Reset = 0000000h]

OUTPUT7MUX16TO31CFG is shown in [Figure 9-69](#) and described in [Table 9-77](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 7

**Figure 9-69. OUTPUT7MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT7 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT7 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT7 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT7 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT7 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT7 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT7 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT7 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT7 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT7 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT7 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT7 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT7 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT7 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT7 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT7 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.15 OUTPUT8MUX0TO15CFG Register (Offset = 1Ch) [Reset = 0000000h]

OUTPUT8MUX0TO15CFG is shown in [Figure 9-70](#) and described in [Table 9-78](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 8

**Figure 9-70. OUTPUT8MUX0TO15CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX15		MUX14		MUX13		MUX12		MUX11		MUX10		MUX9		MUX8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX7		MUX6		MUX5		MUX4		MUX3		MUX2		MUX1		MUX0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX15	R/W	0h	Select Bits for OUTPUT8 Mux15: 00 : Select .0 input for Mux15 01 : Select .1 input for Mux15 10 : Select .2 input for Mux15 11 : Select .3 input for Mux15 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX14	R/W	0h	Select Bits for OUTPUT8 Mux14: 00 : Select .0 input for Mux14 01 : Select .1 input for Mux14 10 : Select .2 input for Mux14 11 : Select .3 input for Mux14 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX13	R/W	0h	Select Bits for OUTPUT8 Mux13: 00 : Select .0 input for Mux13 01 : Select .1 input for Mux13 10 : Select .2 input for Mux13 11 : Select .3 input for Mux13 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX12	R/W	0h	Select Bits for OUTPUT8 Mux12: 00 : Select .0 input for Mux12 01 : Select .1 input for Mux12 10 : Select .2 input for Mux12 11 : Select .3 input for Mux12 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX11	R/W	0h	Select Bits for OUTPUT8 Mux11: 00 : Select .0 input for Mux11 01 : Select .1 input for Mux11 10 : Select .2 input for Mux11 11 : Select .3 input for Mux11 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX10	R/W	0h	Select Bits for OUTPUT8 Mux10: 00 : Select .0 input for Mux10 01 : Select .1 input for Mux10 10 : Select .2 input for Mux10 11 : Select .3 input for Mux10 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX9	R/W	0h	Select Bits for OUTPUT8 Mux9: 00 : Select .0 input for Mux9 01 : Select .1 input for Mux9 10 : Select .2 input for Mux9 11 : Select .3 input for Mux9 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX8	R/W	0h	Select Bits for OUTPUT8 Mux8: 00 : Select .0 input for Mux8 01 : Select .1 input for Mux8 10 : Select .2 input for Mux8 11 : Select .3 input for Mux8 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX7	R/W	0h	Select Bits for OUTPUT8 Mux7: 00 : Select .0 input for Mux7 01 : Select .1 input for Mux7 10 : Select .2 input for Mux7 11 : Select .3 input for Mux7 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX6	R/W	0h	Select Bits for OUTPUT8 Mux6: 00 : Select .0 input for Mux6 01 : Select .1 input for Mux6 10 : Select .2 input for Mux6 11 : Select .3 input for Mux6 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX5	R/W	0h	Select Bits for OUTPUT8 Mux5: 00 : Select .0 input for Mux5 01 : Select .1 input for Mux5 10 : Select .2 input for Mux5 11 : Select .3 input for Mux5 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX4	R/W	0h	Select Bits for OUTPUT8 Mux4: 00 : Select .0 input for Mux4 01 : Select .1 input for Mux4 10 : Select .2 input for Mux4 11 : Select .3 input for Mux4 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX3	R/W	0h	Select Bits for OUTPUT8 Mux3: 00 : Select .0 input for Mux3 01 : Select .1 input for Mux3 10 : Select .2 input for Mux3 11 : Select .3 input for Mux3 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX2	R/W	0h	Select Bits for OUTPUT8 Mux2: 00 : Select .0 input for Mux2 01 : Select .1 input for Mux2 10 : Select .2 input for Mux2 11 : Select .3 input for Mux2 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX1	R/W	0h	Select Bits for OUTPUT8 Mux1: 00 : Select .0 input for Mux1 01 : Select .1 input for Mux1 10 : Select .2 input for Mux1 11 : Select .3 input for Mux1 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX0	R/W	0h	Select Bits for OUTPUT8 Mux0: 00 : Select .0 input for Mux0 01 : Select .1 input for Mux0 10 : Select .2 input for Mux0 11 : Select .3 input for Mux0 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.16 OUTPUT8MUX16TO31CFG Register (Offset = 1Eh) [Reset = 0000000h]

OUTPUT8MUX16TO31CFG is shown in [Figure 9-71](#) and described in [Table 9-79](#).

Return to the [Summary Table](#).

Output X-BAR Mux Configuration for Output 8

**Figure 9-71. OUTPUT8MUX16TO31CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX31		MUX30		MUX29		MUX28		MUX27		MUX26		MUX25		MUX24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX23		MUX22		MUX21		MUX20		MUX19		MUX18		MUX17		MUX16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	MUX31	R/W	0h	Select Bits for OUTPUT8 Mux31: 00 : Select .0 input for Mux31 01 : Select .1 input for Mux31 10 : Select .2 input for Mux31 11 : Select .3 input for Mux31 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29-28	MUX30	R/W	0h	Select Bits for OUTPUT8 Mux30: 00 : Select .0 input for Mux30 01 : Select .1 input for Mux30 10 : Select .2 input for Mux30 11 : Select .3 input for Mux30 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
27-26	MUX29	R/W	0h	Select Bits for OUTPUT8 Mux29: 00 : Select .0 input for Mux29 01 : Select .1 input for Mux29 10 : Select .2 input for Mux29 11 : Select .3 input for Mux29 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25-24	MUX28	R/W	0h	Select Bits for OUTPUT8 Mux28: 00 : Select .0 input for Mux28 01 : Select .1 input for Mux28 10 : Select .2 input for Mux28 11 : Select .3 input for Mux28 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23-22	MUX27	R/W	0h	Select Bits for OUTPUT8 Mux27: 00 : Select .0 input for Mux27 01 : Select .1 input for Mux27 10 : Select .2 input for Mux27 11 : Select .3 input for Mux27 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21-20	MUX26	R/W	0h	Select Bits for OUTPUT8 Mux26: 00 : Select .0 input for Mux26 01 : Select .1 input for Mux26 10 : Select .2 input for Mux26 11 : Select .3 input for Mux26 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19-18	MUX25	R/W	0h	Select Bits for OUTPUT8 Mux25: 00 : Select .0 input for Mux25 01 : Select .1 input for Mux25 10 : Select .2 input for Mux25 11 : Select .3 input for Mux25 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17-16	MUX24	R/W	0h	Select Bits for OUTPUT8 Mux24: 00 : Select .0 input for Mux24 01 : Select .1 input for Mux24 10 : Select .2 input for Mux24 11 : Select .3 input for Mux24 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15-14	MUX23	R/W	0h	Select Bits for OUTPUT8 Mux23: 00 : Select .0 input for Mux23 01 : Select .1 input for Mux23 10 : Select .2 input for Mux23 11 : Select .3 input for Mux23 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13-12	MUX22	R/W	0h	Select Bits for OUTPUT8 Mux22: 00 : Select .0 input for Mux22 01 : Select .1 input for Mux22 10 : Select .2 input for Mux22 11 : Select .3 input for Mux22 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
11-10	MUX21	R/W	0h	Select Bits for OUTPUT8 Mux21: 00 : Select .0 input for Mux21 01 : Select .1 input for Mux21 10 : Select .2 input for Mux21 11 : Select .3 input for Mux21 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9-8	MUX20	R/W	0h	Select Bits for OUTPUT8 Mux20: 00 : Select .0 input for Mux20 01 : Select .1 input for Mux20 10 : Select .2 input for Mux20 11 : Select .3 input for Mux20 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7-6	MUX19	R/W	0h	Select Bits for OUTPUT8 Mux19: 00 : Select .0 input for Mux19 01 : Select .1 input for Mux19 10 : Select .2 input for Mux19 11 : Select .3 input for Mux19 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5-4	MUX18	R/W	0h	Select Bits for OUTPUT8 Mux18: 00 : Select .0 input for Mux18 01 : Select .1 input for Mux18 10 : Select .2 input for Mux18 11 : Select .3 input for Mux18 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	MUX17	R/W	0h	Select Bits for OUTPUT8 Mux17: 00 : Select .0 input for Mux17 01 : Select .1 input for Mux17 10 : Select .2 input for Mux17 11 : Select .3 input for Mux17 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1-0	MUX16	R/W	0h	Select Bits for OUTPUT8 Mux16: 00 : Select .0 input for Mux16 01 : Select .1 input for Mux16 10 : Select .2 input for Mux16 11 : Select .3 input for Mux16 Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.17 OUTPUT1MUXENABLE Register (Offset = 20h) [Reset = 0000000h]

OUTPUT1MUXENABLE is shown in [Figure 9-72](#) and described in [Table 9-80](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 1

**Figure 9-72. OUTPUT1MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT1 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT1 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT1 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.18 OUTPUT2MUXENABLE Register (Offset = 22h) [Reset = 0000000h]

OUTPUT2MUXENABLE is shown in [Figure 9-73](#) and described in [Table 9-81](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 2

**Figure 9-73. OUTPUT2MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT2 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT2 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT2 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.19 OUTPUT3MUXENABLE Register (Offset = 24h) [Reset = 0000000h]

OUTPUT3MUXENABLE is shown in [Figure 9-74](#) and described in [Table 9-82](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 3

**Figure 9-74. OUTPUT3MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT3 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT3 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT3 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.20 OUTPUT4MUXENABLE Register (Offset = 26h) [Reset = 0000000h]

OUTPUT4MUXENABLE is shown in [Figure 9-75](#) and described in [Table 9-83](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 4

**Figure 9-75. OUTPUT4MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT4 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT4 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT4 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.21 OUTPUT5MUXENABLE Register (Offset = 28h) [Reset = 0000000h]

OUTPUT5MUXENABLE is shown in [Figure 9-76](#) and described in [Table 9-84](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 5

**Figure 9-76. OUTPUT5MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT5 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT5 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT5 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.22 OUTPUT6MUXENABLE Register (Offset = 2Ah) [Reset = 0000000h]

OUTPUT6MUXENABLE is shown in [Figure 9-77](#) and described in [Table 9-85](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 6

**Figure 9-77. OUTPUT6MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MUX31	R/W	0h	Selects the output of Mux31 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux31 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux31 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
30	MUX30	R/W	0h	Selects the output of Mux30 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux30 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux30 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
29	MUX29	R/W	0h	Selects the output of Mux29 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux29 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux29 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
28	MUX28	R/W	0h	Selects the output of Mux28 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux28 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux28 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT6 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT6 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT6 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.23 OUTPUT7MUXENABLE Register (Offset = 2Ch) [Reset = 0000000h]

OUTPUT7MUXENABLE is shown in [Figure 9-78](#) and described in [Table 9-86](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 7

**Figure 9-78. OUTPUT7MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-86. 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT7 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT7 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT7 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.24 OUTPUT8MUXENABLE Register (Offset = 2Eh) [Reset = 0000000h]

OUTPUT8MUXENABLE is shown in [Figure 9-79](#) and described in [Table 9-87](#).

Return to the [Summary Table](#).

Output X-BAR Mux Enable for Output 8

**Figure 9-79. OUTPUT8MUXENABLE Register**

31	30	29	28	27	26	25	24
MUX31	MUX30	MUX29	MUX28	MUX27	MUX26	MUX25	MUX24
R/W-0h							
23	22	21	20	19	18	17	16
MUX23	MUX22	MUX21	MUX20	MUX19	MUX18	MUX17	MUX16
R/W-0h							
15	14	13	12	11	10	9	8
MUX15	MUX14	MUX13	MUX12	MUX11	MUX10	MUX9	MUX8
R/W-0h							
7	6	5	4	3	2	1	0
MUX7	MUX6	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0
R/W-0h							

**Table 9-87. 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: CPU1.SYSRSn
26	MUX26	R/W	0h	Selects the output of Mux26 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux26 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux26 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
25	MUX25	R/W	0h	Selects the output of Mux25 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux25 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux25 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
24	MUX24	R/W	0h	Selects the output of Mux24 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux24 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux24 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
23	MUX23	R/W	0h	Selects the output of Mux23 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux23 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux23 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
22	MUX22	R/W	0h	Selects the output of Mux22 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux22 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux22 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
21	MUX21	R/W	0h	Selects the output of Mux21 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux21 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux21 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
20	MUX20	R/W	0h	Selects the output of Mux20 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux20 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux20 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	MUX19	R/W	0h	Selects the output of Mux19 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux19 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux19 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
18	MUX18	R/W	0h	Selects the output of Mux18 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux18 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux18 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
17	MUX17	R/W	0h	Selects the output of Mux17 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux17 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux17 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
16	MUX16	R/W	0h	Selects the output of Mux16 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux16 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux16 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
15	MUX15	R/W	0h	Selects the output of Mux15 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux15 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux15 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
14	MUX14	R/W	0h	Selects the output of Mux14 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux14 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux14 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
13	MUX13	R/W	0h	Selects the output of Mux13 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux13 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux13 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
12	MUX12	R/W	0h	Selects the output of Mux12 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux12 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux12 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	MUX11	R/W	0h	Selects the output of Mux11 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux11 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux11 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
10	MUX10	R/W	0h	Selects the output of Mux10 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux10 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux10 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
9	MUX9	R/W	0h	Selects the output of Mux9 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux9 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux9 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
8	MUX8	R/W	0h	Selects the output of Mux8 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux8 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux8 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
7	MUX7	R/W	0h	Selects the output of Mux7 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux7 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux7 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	MUX6	R/W	0h	Selects the output of Mux6 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux6 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux6 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	MUX5	R/W	0h	Selects the output of Mux5 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux5 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux5 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	MUX4	R/W	0h	Selects the output of Mux4 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux4 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux4 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	MUX3	R/W	0h	Selects the output of Mux3 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux3 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux3 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	MUX2	R/W	0h	Selects the output of Mux2 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux2 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux2 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	MUX1	R/W	0h	Selects the output of Mux1 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux1 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux1 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	MUX0	R/W	0h	Selects the output of mux0 to drive OUTPUT8 of OUTPUT-XBAR 0: Respective output of Mux0 is disabled to drive the OUTPUT8 of OUTPUT-XBAR 1: Respective output of Mux0 is enabled to drive the OUTPUT8 of OUTPUT-XBAR Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.25 OUTPUTLATCH Register (Offset = 30h) [Reset = 0000000h]

OUTPUTLATCH is shown in [Figure 9-80](#) and described in [Table 9-88](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch

**Figure 9-80. OUTPUTLATCH Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 9-88. OUTPUTLATCH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R	0h	Records the OUTPUT8 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
6	OUTPUT7	R	0h	Records the OUTPUT7 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
5	OUTPUT6	R	0h	Records the OUTPUT6 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
4	OUTPUT5	R	0h	Records the OUTPUT5 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn

**Table 9-88. OUTPUTLATCH Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	OUTPUT4	R	0h	Records the OUTPUT4 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
2	OUTPUT3	R	0h	Records the OUTPUT3 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
1	OUTPUT2	R	0h	Records the OUTPUT2 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn
0	OUTPUT1	R	0h	Records the OUTPUT1 of OUTPUT-XBAR. 0: Respective output has not been triggered 1: Respective output is triggered Refer to the Output X-BAR section of this chapter for more details. Note: [1] setting of this bit has priority over clear by software Reset type: CPU1.SYSRSn

### 9.3.5.26 OUTPUTLATCHCLR Register (Offset = 32h) [Reset = 0000000h]

OUTPUTLATCHCLR is shown in [Figure 9-81](#) and described in [Table 9-89](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Clear

**Figure 9-81. OUTPUTLATCHCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0/W1S-0h							

**Table 9-89. OUTPUTLATCHCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R-0/W1S	0h	Clears the Output-Latch for OUTPUT8 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R-0/W1S	0h	Clears the Output-Latch for OUTPUT7 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R-0/W1S	0h	Clears the Output-Latch for OUTPUT6 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R-0/W1S	0h	Clears the Output-Latch for OUTPUT5 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3	OUTPUT4	R-0/W1S	0h	Clears the Output-Latch for OUTPUT4 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-89. OUTPUTLATCHCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	OUTPUT3	R-0/W1S	0h	Clears the Output-Latch for OUTPUT3 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	OUTPUT2	R-0/W1S	0h	Clears the Output-Latch for OUTPUT2 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	OUTPUT1	R-0/W1S	0h	Clears the Output-Latch for OUTPUT1 of OUTPUT-XBAR Writing 1 clears the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.27 OUTPUTLATCHFRC Register (Offset = 34h) [Reset = 0000000h]

OUTPUTLATCHFRC is shown in [Figure 9-82](#) and described in [Table 9-90](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Clear

**Figure 9-82. OUTPUTLATCHFRC Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R-0/W1S-0h							

**Table 9-90. OUTPUTLATCHFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R-0/W1S	0h	Sets the Output-Latch for OUTPUT8 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R-0/W1S	0h	Sets the Output-Latch for OUTPUT7 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R-0/W1S	0h	Sets the Output-Latch for OUTPUT6 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R-0/W1S	0h	Sets the Output-Latch for OUTPUT5 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3	OUTPUT4	R-0/W1S	0h	Sets the Output-Latch for OUTPUT4 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-90. OUTPUTLATCHFRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	OUTPUT3	R-0/W1S	0h	Sets the Output-Latch for OUTPUT3 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
1	OUTPUT2	R-0/W1S	0h	Sets the Output-Latch for OUTPUT2 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
0	OUTPUT1	R-0/W1S	0h	Sets the Output-Latch for OUTPUT1 of OUTPUT-XBAR Writing 1 sets the corresponding output latch bit in the OUTPUTLATCH register Write of 0 has no effect Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.28 OUTPUTLATCHENABLE Register (Offset = 36h) [Reset = 0000000h]

OUTPUTLATCHENABLE is shown in [Figure 9-83](#) and described in [Table 9-91](#).

Return to the [Summary Table](#).

Output X-BAR Output Latch Enable

**Figure 9-83. OUTPUTLATCHENABLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 9-91. OUTPUTLATCHENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R/W	0h	Selects the output latch to drive OUTPUT8 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R/W	0h	Selects the output latch to drive OUTPUT7 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R/W	0h	Selects the output latch to drive OUTPUT6 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R/W	0h	Selects the output latch to drive OUTPUT5 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
3	OUTPUT4	R/W	0h	Selects the output latch to drive OUTPUT4 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
2	OUTPUT3	R/W	0h	Selects the output latch to drive OUTPUT3 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

**Table 9-91. 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. Reset type: CPU1.SYSRSn
0	OUTPUT1	R/W	0h	Selects the output latch to drive OUTPUT1 for OUTPUT-XBAR 0: Output Latch is not selected to driven the respective output 1: Output Latch is selected to drive the respective output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn

### 9.3.5.29 OUTPUTINV Register (Offset = 38h) [Reset = 0000000h]

OUTPUTINV is shown in [Figure 9-84](#) and described in [Table 9-92](#).

Return to the [Summary Table](#).

Output X-BAR Output Inversion

**Figure 9-84. OUTPUTINV Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
OUTPUT8	OUTPUT7	OUTPUT6	OUTPUT5	OUTPUT4	OUTPUT3	OUTPUT2	OUTPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 9-92. OUTPUTINV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-8	RESERVED	R-0	0h	Reserved
7	OUTPUT8	R/W	0h	Selects polarity for OUTPUT8 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
6	OUTPUT7	R/W	0h	Selects polarity for OUTPUT7 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
5	OUTPUT6	R/W	0h	Selects polarity for OUTPUT6 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
4	OUTPUT5	R/W	0h	Selects polarity for OUTPUT5 of OUTPUT-XBAR 0: drives active high output 1: drives active-low output Refer to the Output X-BAR section of this chapter for more details. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

**Table 9-92. 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. Reset type: CPU1.SYSRSn
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. Reset type: CPU1.SYSRSn

### 9.3.5.30 OUTPUTLOCK Register (Offset = 3Eh) [Reset = 0000000h]

OUTPUTLOCK is shown in [Figure 9-85](#) and described in [Table 9-93](#).

Return to the [Summary Table](#).

Output X-BAR Configuration Lock register

**Figure 9-85. OUTPUTLOCK Register**

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

**Table 9-93. OUTPUTLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W1S	0h	Bit-0 of this register can be set only if KEY= 0x5a5a Reset type: CPU1.SYSRSn
15-1	RESERVED	R-0	0h	Reserved
0	LOCK	R/WOnce	0h	Locks the configuration for OUTPUT-XBAR. Once the configuration is locked, writes to the below registers for OUTPUT-XBAR is blocked. Registers Affected by the LOCK mechanism: OUTPUT-XBAROUTyMUX0TO15CFG OUTPUT-XBAROUTyMUX16TO31CFG OUTPUT-XBAROUTyMUXENABLE OUTPUT-XBAROUTLATENABLE OUTPUT-XBAROUTINV 0: Writes to the above registers are allowed 1: Writes to the above registers are blocked Note: [1] LOCK mechanism only applies to writes. Reads are never blocked. Reset type: CPU1.SYSRSn

### 9.3.6 Register to Driverlib Function Mapping

#### 9.3.6.1 INPUTXBAR Registers to Driverlib Functions

**Table 9-94. INPUTXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>INPUT1SELECT</b>	
xbar.h	XBAR_setInputPin
<b>INPUT2SELECT</b>	
-	See INPUT1SELECT
<b>INPUT3SELECT</b>	
-	See INPUT1SELECT
<b>INPUT4SELECT</b>	
-	See INPUT1SELECT
<b>INPUT5SELECT</b>	
-	See INPUT1SELECT
<b>INPUT6SELECT</b>	
-	See INPUT1SELECT
<b>INPUT7SELECT</b>	
-	See INPUT1SELECT
<b>INPUT8SELECT</b>	
-	See INPUT1SELECT
<b>INPUT9SELECT</b>	
-	See INPUT1SELECT
<b>INPUT10SELECT</b>	
-	See INPUT1SELECT
<b>INPUT11SELECT</b>	
-	See INPUT1SELECT
<b>INPUT12SELECT</b>	
-	See INPUT1SELECT
<b>INPUT13SELECT</b>	
-	See INPUT1SELECT
<b>INPUT14SELECT</b>	
-	See INPUT1SELECT
<b>INPUT15SELECT</b>	
-	See INPUT1SELECT
<b>INPUT16SELECT</b>	
-	See INPUT1SELECT
<b>INPUTSELECTLOCK</b>	
xbar.h	XBAR_lockInput

#### 9.3.6.2 XBAR Registers to Driverlib Functions

**Table 9-95. XBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>FLG1</b>	
xbar.c	XBAR_getInputFlagStatus
<b>FLG2</b>	
xbar.c	XBAR_getInputFlagStatus

**Table 9-95. XBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>FLG3</b>	
xbar.c	XBAR_getInputFlagStatus
<b>FLG4</b>	
xbar.c	XBAR_getInputFlagStatus
<b>CLR1</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR2</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR3</b>	
xbar.c	XBAR_clearInputFlag
<b>CLR4</b>	
xbar.c	XBAR_clearInputFlag

**9.3.6.3 EPWMXBAR Registers to Driverlib Functions****Table 9-96. EPWMXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>TRIP4MUX0TO15CFG</b>	
xbar.c	XBAR_setEPWMMuxConfig
<b>TRIP4MUX16TO31CFG</b>	
xbar.c	XBAR_setEPWMMuxConfig
<b>TRIP5MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP5MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP7MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP7MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP8MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP8MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP9MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP9MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP10MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP10MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP11MUX0TO15CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP11MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP12MUX0TO15CFG</b>	

**Table 9-96. EPWMXBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See TRIP4MUX0TO15CFG
<b>TRIP12MUX16TO31CFG</b>	
-	See TRIP4MUX0TO15CFG
<b>TRIP4MUXENABLE</b>	
xbar.h	XBAR_enableEPWMMux
xbar.h	XBAR_disableEPWMMux
<b>TRIP5MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP7MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP8MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP9MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP10MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP11MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIP12MUXENABLE</b>	
-	See TRIP4MUXENABLE
<b>TRIPOUTINV</b>	
xbar.h	XBAR_invertEPWMSignal
<b>TRIPLOCK</b>	
xbar.h	XBAR_lockEPWM

### 9.3.6.4 OUTPUTXBAR Registers to Driverlib Functions

**Table 9-97. OUTPUTXBAR Registers to Driverlib Functions**

File	Driverlib Function
<b>OUTPUT1MUX0TO15CFG</b>	
xbar.c	XBAR_setOutputMuxConfig
<b>OUTPUT1MUX16TO31CFG</b>	
-	
<b>OUTPUT2MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT2MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT3MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT3MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT4MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT4MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT5MUX0TO15CFG</b>	

**Table 9-97. OUTPUTXBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT5MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT6MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT6MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT7MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT7MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT8MUX0TO15CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT8MUX16TO31CFG</b>	
-	See OUTPUT1MUX0TO15CFG
<b>OUTPUT1MUXENABLE</b>	
xbar.h	XBAR_enableOutputMux
xbar.h	XBAR_disableOutputMux
<b>OUTPUT2MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT3MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT4MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT5MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT6MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT7MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUT8MUXENABLE</b>	
-	See OUTPUT1MUXENABLE
<b>OUTPUTLATCH</b>	
xbar.h	XBAR_setOutputLatchMode
xbar.h	XBAR_getOutputLatchStatus
xbar.h	XBAR_clearOutputLatch
xbar.h	XBAR_forceOutputLatch
<b>OUTPUTLATCHCLR</b>	
xbar.h	XBAR_clearOutputLatch
<b>OUTPUTLATCHFRC</b>	
xbar.h	XBAR_forceOutputLatch
<b>OUTPUTLATCHENABLE</b>	
xbar.h	XBAR_setOutputLatchMode
<b>OUTPUTINV</b>	
xbar.h	XBAR_invertOutputSignal

**Table 9-97. OUTPUTXBAR Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>OUTPUTLOCK</b>	
xbar.h	XBAR_lockOutput

This page intentionally left blank.

Chapter 10  
**Analog Subsystem**

---



The analog subsystem module is described in this chapter.

<b>10.1 Introduction</b> .....	<b>974</b>
<b>10.2 Optimizing Power-Up Time</b> .....	<b>976</b>
<b>10.3 Digital Inputs on ADC Pins (AIOs)</b> .....	<b>977</b>
<b>10.4 Digital Inputs and Outputs on ADC Pins (AGPIOs)</b> .....	<b>977</b>
<b>10.5 Analog Pins and Internal Connections</b> .....	<b>978</b>
<b>10.6 Analog Subsystem Registers</b> .....	<b>980</b>

## 10.1 Introduction

The analog modules on this device include the Analog-to-Digital Converter (ADC), Temperature Sensor, Comparator Subsystem (CMPSS), and Lite Comparator Subsystem variant (CMPSS\_LITE).

### 10.1.1 Features

The analog subsystem has the following features:

- Flexible voltage references
  - The ADCs are referenced to VREFHI and VSSA pins
    - VREFHI pin voltage can be driven in externally or can be generated by an internal bandgap voltage reference. For 32RHB package, VREFHI is internally tied to VDDA.
    - The internal voltage reference range can be selected to be 0V to 3.3V or 0V to 2.5V
  - The comparator DACs are referenced to VDDA and VSSA
- Flexible pin usage
  - Comparator subsystem inputs and digital inputs (AIOs)/outputs (AGPIOs) are multiplexed with ADC inputs
  - Comparator DAC can optionally be buffered and brought out to a multiplexed ADC pin (mutually exclusive with use of CMPSS compare functions)
  - Internal connection to  $V_{REFLO}$  on all ADCs for offset self-calibration

### 10.1.2 Block Diagram

The following analog subsystem block diagrams show the connections between the different integrated analog modules to the device pins. These pins fall into two categories: analog module inputs/outputs and reference pins.

The analog pins are organized into analog groups around the CMPSS module. 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.

Some analog pins support digital functionality through muxed AIOs and AGPIOs. AIOs support only digital input functionality while AGPIOs support full digital input and output functionality.

The following notes apply to all packages:

- Not all analog pins are available on all devices. See the device data sheet to determine which pins are available.
- See the device data sheet to determine the allowable voltage range for VREFHI and VREFLO.
- An external capacitor is required on the VREFHI pins. See the device data sheet for the specific value required.

[Figure 10-2](#) shows how each analog group is structured. [Section 10.5](#) lists the analog pins and internal connections.

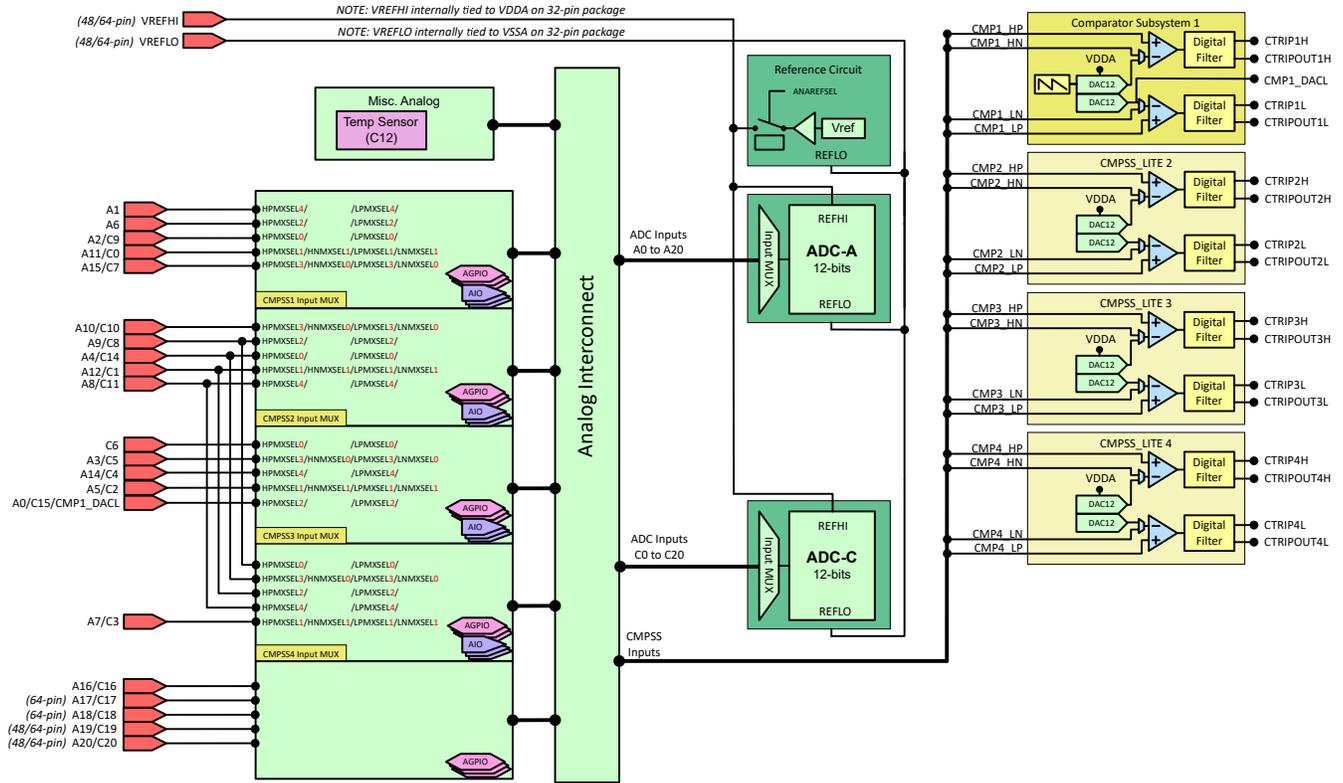
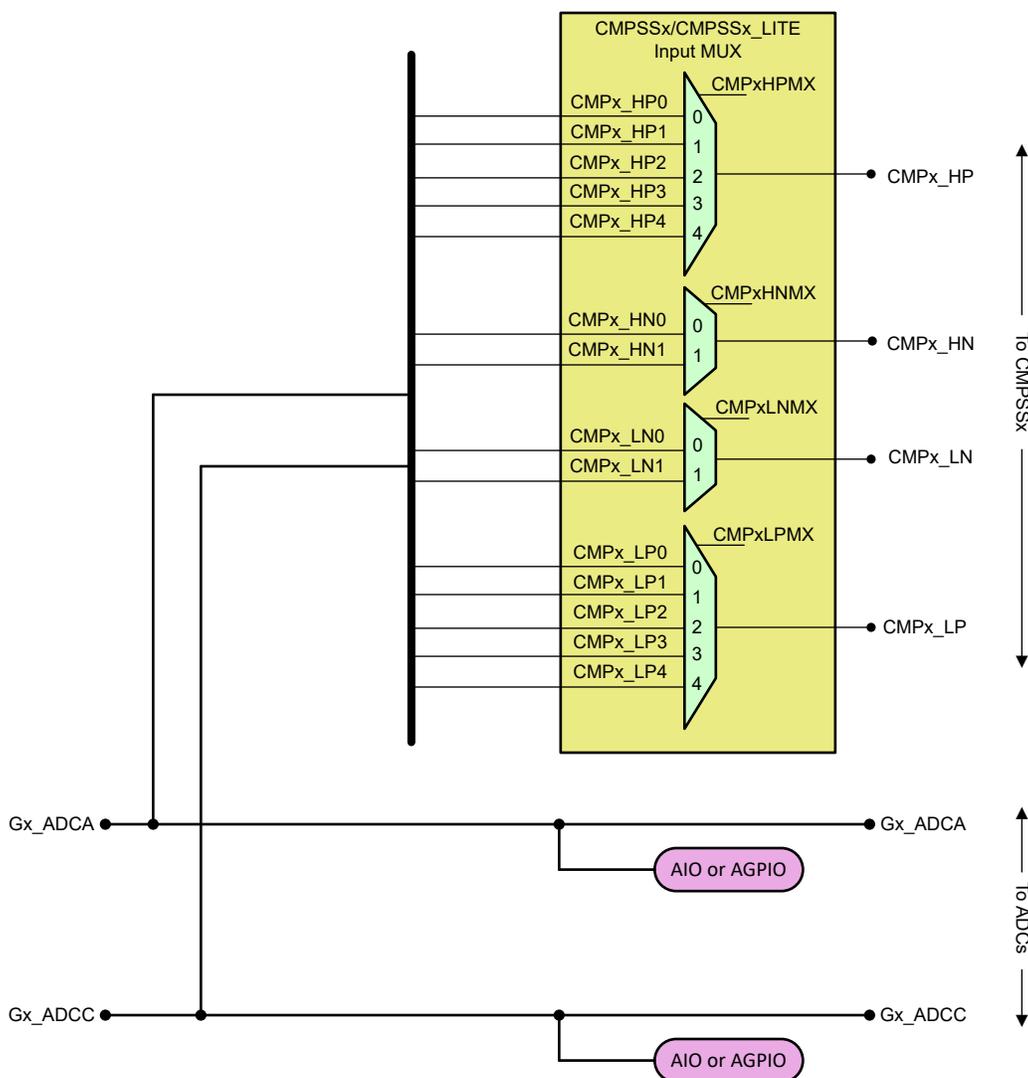


Figure 10-1. Analog Subsystem Block Diagram



Note: AIOs support digital input mode only.

Figure 10-2. Analog Group Connections

## 10.2 Optimizing Power-Up Time

The analog-to-digital converters (ADC) and buffered digital-to-analog converters (DAC) share a common reference circuit. If needed, an application using one or more of these modules can optimize power-up time by taking advantage of the shared reference. Once one of the modules using the shared reference has been initialized in internal reference mode, the power-up time for subsequent modules can be optimized by subtracting the reference power-up time from the minimum power-up time requirement.

For instance, if ADCA requires  $t_{ADCPUIINT}$  to power up in internal reference mode, and  $t_{ADCPUEXT}$  to power up in external reference mode, the application does not need to wait  $t_{ADCPUIINT}$  to power up a second ADC instance such as ADCC in internal reference mode. In this case, the application can simply wait for  $t_{ADCPUEXT}$  after powering up ADCC, even though both ADCs are used in internal reference mode. In the same scenario, if the application wished to use DACA in internal reference mode, the required wait time after power-up is  $t_{DACPUEXT}$ , not the longer  $t_{ADCPUIINT}$ .

There is also a wait time associated with power-up in internal reference mode when switching between 2.5V and 3.3V range. See the device data sheet for wait time values.

### 10.3 Digital Inputs on ADC Pins (AIOs)

Some GPIOs are multiplexed with analog pins and only have digital input functionality. These are also referred to as AIOs. Pins with only an AIO option on this port can only function in input mode. See the device data sheet for list of AIO signals. By default, these pins function as analog pins and the GPIOs are in a high-impedance state. The GPyAMSEL register is used to configure these pins for digital or analog operation.

---

#### Note

If digital signals with sharp edges (high dv/dt) are connected to the AIOs, cross-talk can occur with adjacent analog signals. Therefore, limit the edge rate of signals connected to AIOs if adjacent channels are being used for analog functions.

---

### 10.4 Digital Inputs and Outputs on ADC Pins (AGPIOs)

Some GPIOs are multiplexed with analog pins and have digital input and output functionality. These are also referred to as AGPIOs. Unlike AIOs, AGPIOs have full input and output capability.

By default, the AGPIOs are not connected and must be configured. [Table 10-1](#) shows how to configure the AGPIOs. To enable the analog functionality, set the register AGPIOTRLx from analog subsystem. To enable the digital functionality, set the register GPxAMSEL from the *General-Purpose Input/Output (GPIO)* chapter.

**Table 10-1. AGPIO Configuration**

AGPIOTRLx.GPIOy (Default = 0)	GPxAMSEL.GPIOy (Default = 1)	Pin Connected To:	
		ADC	GPIOy
0	0	-	Yes
<b>0</b>	<b>1</b>	- <sup>(1)</sup>	- <sup>(1)</sup>
1	0	-	Yes
1	1	Yes	-

(1) By default there are no signals connected to AGPIO pins. One of the other rows in the table must be chosen for pin functionality.

---

#### Note

If digital signals with sharp edges (high dv/dt) are connected to the AGPIOs, cross-talk can occur with adjacent analog signals. The user must therefore limit the edge rate of signals connected to AGPIOs, if adjacent channels are being used for analog functions.

---

## 10.5 Analog Pins and Internal Connections

**Table 10-2. Analog Pins and Internal Connections**

Pin Name	Pins/Package				ADC		DAC	Comparator Subsystem (Mux)				AIO Input/ GPIO
	64 QFP <sup>(5)</sup>	48 QFP	48 QFN	32 QFN	A	C		High Positive	High Negative	Low Positive	Low Negative	
VREFHI	16	12	12	.(4)								
VREFLO	17	13	13	.(4)	A13	C13						
<b>Analog Group 1</b>								<b>CMP1</b>				
A6	6	4 <sup>(1)</sup>	4 <sup>(1)</sup>	2 <sup>(1)</sup>	A6	-		CMP1 (HPMXSEL=2)		CMP1 (LPMXSEL=2)		GPIO228 <sup>(3)</sup>
A2/C9	9	6	6	4	A2	C9		CMP1 (HPMXSEL=0)		CMP1 (LPMXSEL=0)		GPIO224 <sup>(3)</sup>
A15/C7	10	7 <sup>(1)</sup>	7 <sup>(1)</sup>	5 <sup>(1)</sup>	A15	C7		CMP1 (HPMXSEL=3)	CMP1 (HNMXSEL=0)	CMP1 (LPMXSEL=3)	CMP1 (LNMXSEL=0)	AIO233
A11/C0	12	8	8	6 <sup>(1)</sup>	A11	C0		CMP1 (HPMXSEL=1)	CMP1 (HNMXSEL=1)	CMP1 (LPMXSEL=1)	CMP1 (LNMXSEL=1)	AIO237
A1	14	10	10	7 <sup>(1)</sup>	A1	-		CMP1 (HPMXSEL=4)		CMP1 (LPMXSEL=4)		AIO232
<b>Analog Group 2</b>								<b>CMP2</b>				
A10/C10	25	21	21	13 <sup>(1)</sup>	A10	C10		CMP2 (HPMXSEL=3)	CMP2 (HNMXSEL=0)	CMP2 (LPMXSEL=3)	CMP2 (LNMXSEL=0)	GPIO230 <sup>(3)</sup>
<b>Analog Group 3</b>								<b>CMP3</b>				
C6	7	4 <sup>(1)</sup>	4 <sup>(1)</sup>	2 <sup>(1)</sup>	-	C6		CMP3 (HPMXSEL=0)		CMP3 (LPMXSEL=0)		GPIO226 <sup>(3)</sup>
A3/C5	8	5	5	3	A3	C5		CMP3 (HPMXSEL=3)	CMP3 (HNMXSEL=0)	CMP3 (LPMXSEL=3)	CMP3 (LNMXSEL=0)	GPIO242 <sup>(3)</sup>
A14/C4	11	7 <sup>(1)</sup>	7 <sup>(1)</sup>	5 <sup>(1)</sup>	A14	C4		CMP3 (HPMXSEL=4)		CMP3 (LPMXSEL=4)		AIO239
A5/C2	13	9	9	6 <sup>(1)</sup>	A5	C2		CMP3 (HPMXSEL=1)	CMP3 (HNMXSEL=1)	CMP3 (LPMXSEL=1)	CMP3 (LNMXSEL=1)	AIO244
A0/C15/CMP1_DACL	15	11	11	7 <sup>(1)</sup>	A0	C15	CMP1_DACL	CMP3 (HPMXSEL=2)		CMP3 (LPMXSEL=2)		AIO231
<b>Analog Group 4</b>								<b>CMP4</b>				
A7/C3	19	15	15	8 <sup>(1)</sup>	A7	C3		CMP4 (HPMXSEL=1)	CMP4 (HNMXSEL=1)	CMP4 (LPMXSEL=1)	CMP4 (LNMXSEL=1)	AIO245
<b>Combined Analog Group 2/4</b>								<b>CMP2/4</b>				
A12/C1	18	14	14	8 <sup>(1)</sup>	A12	C1		CMP2 (HPMXSEL=1) CMP4 (HPMXSEL=2)	CMP2 (HNMXSEL=1)	CMP2 (LPMXSEL=1) CMP4 (LPMXSEL=2)	CMP2 (LNMXSEL=1)	AIO238
A8/C11	20	16	16	9	A8	C11		CMP2 (HPMXSEL=4) CMP4 (HPMXSEL=4)		CMP2 (LPMXSEL=4) CMP4 (LPMXSEL=4)		AIO241
A4/C14	23	19	19	12	A4	C14		CMP2 (HPMXSEL=0) CMP4 (HPMXSEL=3)	CMP4 (HNMXSEL=0)	CMP2 (LPMXSEL=0) CMP4 (LPMXSEL=3)	CMP4 (LNMXSEL=0)	AIO225
A9/C8	24	20	20	13 <sup>(1)</sup>	A9	C8		CMP2 (HPMXSEL=2) CMP4 (HPMXSEL=0)		CMP2 (LPMXSEL=2) CMP4 (LPMXSEL=0)		GPIO227 <sup>(3)</sup>
<b>Other Analog</b>												
TempSensor <sup>(2)</sup>	-	-	-	-	-	C12		CMP2 (HPMXSEL=5)				
A16/C16	2	2	2	32	A16	C16						GPIO28 <sup>(3)</sup>
A17/C17	27	-	-	-	A17	C17						GPIO20 <sup>(3)</sup>
A18/C18	28	-	-	-	A18	C18						GPIO21 <sup>(3)</sup>
A19/C19	29	23	22	-	A19	C19						GPIO13 <sup>(3)</sup>

**Table 10-2. Analog Pins and Internal Connections (continued)**

Pin Name	Pins/Package				ADC		DAC	Comparator Subsystem (Mux)				AIO Input/ GPIO
	64 QFP <sup>(5)</sup>	48 QFP	48 QFN	32 QFN	A	C		High Positive	High Negative	Low Positive	Low Negative	
A20/C20	30	24	23	-	A20	C20						GPIO12 <sup>(3)</sup>

- (1) Signal is bonded together with another signal as a single pin on this package.
- (2) Internal connection only; does not come to a device pin.
- (3) The GPIOs on these analog pins support full digital input and output functionality and are referred to as AGPIOs. By default, the AGPIOs are unconnected; that is, the analog and digital functions are both disabled. For configuration details, see the *Digital Inputs and Outputs on ADC Pins (AGPIOs)* section.
- (4) On 32 RHB package, VREFHI is internally connected to VDDA and VREFLO is internally connected to VSSA.
- (5) Column applies to both 64 PM and 64 PM with VREGENZ (VPM) variants.

**Table 10-3. Analog Signal Descriptions**

Signal Name	Description
AIOx	Digital input on ADC pin
Ax	ADC A Input
Cx	ADC C Input
CMPx_HNy	Comparator subsystem high comparator negative input
CMPx_HPy	Comparator subsystem high comparator positive input
CMPx_LNy	Comparator subsystem low comparator negative input
CMPx_LPy	Comparator subsystem low comparator positive input
CMPx_DACL	DAC output from the lower CMPSS DAC (can be brought to an external pin)
TempSensor	Internal temperature sensor

**Table 10-4. Reference Summary**

Module	Reference Option	Configured Where?	Register	Driverlib Function	Notes
ADC	Internal or External	Analog System	AnalogSubsysRegs. ANAREFCTL.bit. ANAREFxSEL	ADC_setVREF	Both options require use of the VREFHI pin.
	3.3V or 2.5V Internal Reference Range	Analog System	AnalogSubsysRegs. ANAREFCTL.bit. ANAREFx2P5SEL	ADC_setVREF	Only applicable when using internal reference mode.
CMPSS DACs	VDDA	CMPSS Module	Not configurable		

## 10.6 Analog Subsystem Registers

This section describes the Analog Subsystem Registers.

### 10.6.1 ASBSYS Base Address Table

**Table 10-5. ASBSYS Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
AnalogSubsysRegs	ANALOG_SUBSYS_REGS	ANALOGSUBSYS_BASE	0x0005_D700	YES

## 10.6.2 ANALOG\_SUBSYS\_REGS Registers

Table 10-6 lists the memory-mapped registers for the ANALOG\_SUBSYS\_REGS registers. All register offset addresses not listed in Table 10-6 should be considered as reserved locations and the register contents should not be modified.

**Table 10-6. ANALOG\_SUBSYS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
24h	EXTROSCCSR1	ExtR Oscillator Status Register	EALLOW	<a href="#">Go</a>
4Ah	INTERNALTESTCTL	INTERNALTEST Node Control Register	EALLOW	<a href="#">Go</a>
5Eh	CONFIGLOCK	Lock Register for all the config registers.	EALLOW	<a href="#">Go</a>
60h	TSNSCTL	Temperature Sensor Control Register	EALLOW	<a href="#">Go</a>
68h	ANAREFCTL	Analog Reference Control Register. This register is not configurable for 32QFN package	EALLOW	<a href="#">Go</a>
70h	VMONCTL	Voltage Monitor Control Register	EALLOW	<a href="#">Go</a>
82h	CMPPHMXSEL	Bits to select one of the many sources on CompHP inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
84h	CMPLPMXSEL	Bits to select one of the many sources on ComplP inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
86h	CMPHNMXSEL	Bits to select one of the many sources on CompHN inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
87h	CMPLNMXSEL	Bits to select one of the many sources on ComplN inputs. Refer to Pimux diagram for details.	EALLOW	<a href="#">Go</a>
88h	ADCDALOOPBACK	Enable loopback from DAC to ADCs		<a href="#">Go</a>
8Bh	CMPSSCTL	CMPSS Control Register	EALLOW	<a href="#">Go</a>
8Eh	LOCK	Lock Register	EALLOW	<a href="#">Go</a>
10Ah	AGPIOCTRLA	AGPIO Control Register	EALLOW	<a href="#">Go</a>
118h	AGPIOCTRLH	AGPIO Control Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 10-7 shows the codes that are used for access types in this section.

**Table 10-7. ANALOG\_SUBSYS\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
WSonce	WSonce	Write Set once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		

**Table 10-7. ANALOG\_SUBSYS\_REGS Access Type Codes (continued)**

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

### 10.6.2.1 EXTROSCCSR1 Register (Offset = 24h) [Reset = 0000000h]

EXTROSCCSR1 is shown in [Figure 10-3](#) and described in [Table 10-8](#).

Return to the [Summary Table](#).

ExtR Oscillator Status Register

**Figure 10-3. EXTROSCCSR1 Register**

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

**Table 10-8. EXTROSCCSR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	OSCSTATUS	R	0h	Running status of ExtR. Value is 0xE7 when ExtR enable has completed successfully. 0xE7: ExtR enable completed successfully. OTHER VALUES: ExtR enable incomplete. Reset type: XRSn
23-16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13-12	RESERVED	R/W	0h	Reserved
11-9	RESERVED	R/W	0h	Reserved
8-7	RESERVED	R/W	0h	Reserved
6-4	RESERVED	R/W	0h	Reserved
3-2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

### 10.6.2.2 INTERNALTESTCTL Register (Offset = 4Ah) [Reset = 0000000h]

INTERNALTESTCTL is shown in [Figure 10-4](#) and described in [Table 10-9](#).

Return to the [Summary Table](#).

INTERNALTEST Node Control Register

**Figure 10-4. INTERNALTESTCTL 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				TESTSEL			
R/W-0h				R/W-0h			

**Table 10-9. INTERNALTESTCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Write Key. Writes to this register must include the value 0xA5A5 in the KEY bit field to take effect. Otherwise the register will remain as it was prior to the write attempt. Reads will return a 0. Reset type: SYSRSn
15-8	RESERVED	R-0	0h	Reserved
7-5	RESERVED	R/W	0h	Reserved

**Table 10-9. INTERNALTESTCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	TESTSEL	R/W	0h	<p>Test Select. This bit field defines which internal node, if any, is selected to come out on the INTERNALTEST node connected to the ADC.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No internal connection            1h (R/W) = Core VDD (1.2V) voltage            2h (R/W) = VREFLO pin voltage            3h (R/W) = Reserved            4h (R/W) = CMPSS1 High DAC output (6-bit)            5h (R/W) = CMPSS1 Low DAC output (6-bit)            6h (R/W) = CMPSS2 High DAC output (6-bit)            7h (R/W) = CMPSS2 Low DAC output (6-bit)            8h (R/W) = CMPSS3 High DAC output (6-bit)            9h (R/W) = CMPSS3 Low DAC output (6-bit)            Ah (R/W) = CMPSS4 High DAC output (6-bit)            Bh (R/W) = CMPSS4 Low DAC output (6-bit)            Ch (R/W) = VDDA voltage            Dh (R/W) = VSSA - Analog ground pin            Eh (R/W) = Reserved            Fh (R/W) = Reserved            10h (R/W) = Reserved            11h (R/W) = Reserved            12h (R/W) = Reserved            13h (R/W) = All ADCs are placed in gain calibration mode. 0.9*VREFHI pin voltage is sampled by all ADCs through INTERNALTEST mux output, overriding CHSEL setting.            14h (R/W) = Reserved            15h (R/W) = Reserved            16h (R/W) = Reserved            17h (R/W) = Reserved            18h (R/W) = Reserved            19h (R/W) = Reserved            1Ah (R/W) = Reserved            1Bh (R/W) = Reserved            1Ch (R/W) = Reserved            1Dh (R/W) = Reserved            1Eh (R/W) = Reserved            1Fh (R/W) = Reserved</p>

### 10.6.2.3 CONFIGLOCK Register (Offset = 5Eh) [Reset = 0000000h]

CONFIGLOCK is shown in [Figure 10-5](#) and described in [Table 10-10](#).

Return to the [Summary Table](#).

Lock Register for all the config registers.

**Figure 10-5. CONFIGLOCK 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	AGPIOCTRL	RESERVED	RESERVED	RESERVED
R-0-0h		R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 10-10. CONFIGLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5	RESERVED	R/WOnce	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	AGPIOCTRL	R/WOnce	0h	Locks all AGPIOCTRL Register. Setting this bit will disable any future writes to this register. This bit can only be cleared by a reset. Reset type: SYSRSn
2	RESERVED	R/WOnce	0h	Reserved
1	RESERVED	R/WOnce	0h	Reserved
0	RESERVED	R/WOnce	0h	Reserved

### 10.6.2.4 TSN\_SCTL Register (Offset = 60h) [Reset = 0000h]

TSN\_SCTL is shown in [Figure 10-6](#) and described in [Table 10-11](#).

Return to the [Summary Table](#).

Temperature Sensor Control Register

**Figure 10-6. TSN\_SCTL Register**

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

**Table 10-11. TSN\_SCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R-0	0h	Reserved
0	ENABLE	R/W	0h	Temperature Sensor Enable. This bit enables the temperature sensor output to the ADC. 0 Disabled 1 Enabled Reset type: SYSRSn

### 10.6.2.5 ANAREFCTL Register (Offset = 68h) [Reset = 000Fh]

ANAREFCTL is shown in [Figure 10-7](#) and described in [Table 10-12](#).

Return to the [Summary Table](#).

Analog Reference Control Register. This register is not configurable for 32QFN package

**Figure 10-7. ANAREFCTL Register**

15	14	13	12	11	10	9	8
ANAREFSEL_SUP_OVERRIDE	RESERVED				RESERVED	RESERVED	ANAREF2P5SEL
R/W-0h	R-0-0h				R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED					RESERVED	RESERVED	ANAREFSEL
R-0-1h					R/W-1h	R/W-1h	R/W-1h

**Table 10-12. ANAREFCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	ANAREFSEL_SUP_OVERRIDE	R/W	0h	This configuration determines (overrides) the external reference selection for ADC 1: VDDA/VSSA will be used as the external reference for ADC. In this mode ANAREFSEL must be set to '1' to select external reference mode. 0: VREFHI/VREFLO will be used as the reference pins for the ADC. In this mode ANAREFSEL can be either '0'(internal reference mode) or '1'(external reference mode) Reset type: XRSn
14-11	RESERVED	R-0	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	ANAREF2P5SEL	R/W	0h	Analog reference A 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. 0 Internal 1.65V reference mode (3.3V reference range) 1 Internal 2.5V reference mode (2.5V reference range) Reset type: XRSn
7-3	RESERVED	R-0	1h	Reserved
2	RESERVED	R/W	1h	Reserved
1	RESERVED	R/W	1h	Reserved
0	ANAREFSEL	R/W	1h	Analog reference mode select. This bit selects whether the VREFHI pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). 0 Internal reference mode 1 External reference mode Reset type: XRSn

### 10.6.2.6 VMONCTL Register (Offset = 70h) [Reset = 0000h]

VMONCTL is shown in [Figure 10-8](#) and described in [Table 10-13](#).

Return to the [Summary Table](#).

Voltage Monitor Control Register

**Figure 10-8. VMONCTL Register**

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

**Table 10-13. VMONCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R-0	0h	Reserved
8	BORLVMONDIS	R/W	0h	BORL disable on VDDIO. 0 BORL is enabled on VDDIO, i.e BOR circuit will be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO. 1 BORL is disabled on VDDIO, i.e BOR circuit will not be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO. Reset type: SYSRSn
7-2	RESERVED	R-0	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

### 10.6.2.7 CMPHPMXSEL Register (Offset = 82h) [Reset = 0000000h]

CMPHPMXSEL is shown in [Figure 10-9](#) and described in [Table 10-14](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CompHP inputs. Refer to Pimux diagram for details.

**Figure 10-9. CMPHPMXSEL 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/W-0h			R/W-0h		
15	14	13	12	11	10	9	8
RESERVED	RESERVED			CMP4HPMXSEL		CMP3HPMXSEL	
R-0-0h		R/W-0h			R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
CMP3HPMXSEL		CMP2HPMXSEL			CMP1HPMXSEL		
R/W-0h		R/W-0h			R/W-0h		

**Table 10-14. CMPHPMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R-0	0h	Reserved
21-19	RESERVED	R/W	0h	Reserved
18-16	RESERVED	R/W	0h	Reserved
15	RESERVED	R-0	0h	Reserved
14-12	RESERVED	R/W	0h	Reserved
11-9	CMP4HPMXSEL	R/W	0h	CMP4HPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
8-6	CMP3HPMXSEL	R/W	0h	CMP3HPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
5-3	CMP2HPMXSEL	R/W	0h	CMP2HPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 5 are valid, rest are reserved Reset type: XRSn
2-0	CMP1HPMXSEL	R/W	0h	CMP1HPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn

### 10.6.2.8 CMPLPMXSEL Register (Offset = 84h) [Reset = 0000000h]

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

Return to the [Summary Table](#).

Bits to select one of the many sources on CompLP inputs. Refer to Pimux diagram for details.

**Figure 10-10. CMPLPMXSEL 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/W-0h			R/W-0h		
15	14	13	12	11	10	9	8
RESERVED	RESERVED			CMP4LPMXSEL		CMP3LPMXSEL	
R-0-0h		R/W-0h			R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
CMP3LPMXSEL		CMP2LPMXSEL			CMP1LPMXSEL		
R/W-0h		R/W-0h			R/W-0h		

**Table 10-15. CMPLPMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R-0	0h	Reserved
21-19	RESERVED	R/W	0h	Reserved
18-16	RESERVED	R/W	0h	Reserved
15	RESERVED	R-0	0h	Reserved
14-12	RESERVED	R/W	0h	Reserved
11-9	CMP4LPMXSEL	R/W	0h	CMP4LPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
8-6	CMP3LPMXSEL	R/W	0h	CMP3LPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
5-3	CMP2LPMXSEL	R/W	0h	CMP2LPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn
2-0	CMP1LPMXSEL	R/W	0h	CMP1LPMXSEL bits, Refer to the Analog Subsystem chapter Note: Only values 0 to 4 are valid, rest are reserved Reset type: XRSn

### 10.6.2.9 CMPHNMXSEL Register (Offset = 86h) [Reset = 0000h]

CMPHNMXSEL is shown in [Figure 10-11](#) and described in [Table 10-16](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CompHN inputs. Refer to Pimux diagram for details.

**Figure 10-11. CMPHNMXSEL Register**

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	CMP4HNMXSE L	CMP3HNMXSE L	CMP2HNMXSE L	CMP1HNMXSE L						
R-0-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 10-16. CMPHNMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	CMP4HNMXSEL	R/W	0h	CMP4HNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
2	CMP3HNMXSEL	R/W	0h	CMP3HNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
1	CMP2HNMXSEL	R/W	0h	CMP2HNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
0	CMP1HNMXSEL	R/W	0h	CMP1HNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn

### 10.6.2.10 CMPLNMXSEL Register (Offset = 87h) [Reset = 0000h]

CMPLNMXSEL is shown in [Figure 10-12](#) and described in [Table 10-17](#).

Return to the [Summary Table](#).

Bits to select one of the many sources on CompLN inputs. Refer to Pimux diagram for details.

**Figure 10-12. CMPLNMXSEL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	RESERVED	RESERVED	CMP4LNMXSEL	CMP3LNMXSEL	CMP2LNMXSEL	CMP1LNMXSEL
R-0-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 10-17. CMPLNMXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	CMP4LNMXSEL	R/W	0h	CMP4LNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
2	CMP3LNMXSEL	R/W	0h	CMP3LNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
1	CMP2LNMXSEL	R/W	0h	CMP2LNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn
0	CMP1LNMXSEL	R/W	0h	CMP1LNMXSEL bits, Refer to the Analog Subsystem chapter Reset type: XRSn

### 10.6.2.11 ADCDACLOOPBACK Register (Offset = 88h) [Reset = 0000000h]

ADCDACLOOPBACK is shown in [Figure 10-13](#) and described in [Table 10-18](#).

Return to the [Summary Table](#).

Enable loopback from DAC to ADCs

**Figure 10-13. ADCDACLOOPBACK 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					ENLB2ADCC	RESERVED	ENLB2ADCA
R-0-0h					R/W-0h	R/W-0h	R/W-0h

**Table 10-18. ADCDACLOOPBACK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Write Key. Writes to this register must include the value 0xA5A5 in the KEY bit field to take effect. Otherwise the register will remain as it was prior to the write attempt. Reads will return a 0. Reset type: XRSn
15-3	RESERVED	R-0	0h	Reserved
2	ENLB2ADCC	R/W	0h	1 Loops back COMPDACA output to ADCC. 0 Loop back is broken. Note: Setting this bit to 1, will override the CHSEL specification for the ADC. ADC would sample COMPDACA output irrespective of the value of CHSEL. Reset type: XRSn
1	RESERVED	R/W	0h	Reserved
0	ENLB2ADCA	R/W	0h	1 Loops back COMPDACA output to ADCA. 0 Loop back is broken. Note: Setting this bit to 1, will override the CHSEL specification for the ADC. ADC would sample COMPDACA output irrespective of the value of CHSEL. Reset type: XRSn

### 10.6.2.12 CMPSSCTL Register (Offset = 8Bh) [Reset = 0000h]

CMPSSCTL is shown in [Figure 10-14](#) and described in [Table 10-19](#).

Return to the [Summary Table](#).

CMPSS Control Register

**Figure 10-14. CMPSSCTL Register**

15	14	13	12	11	10	9	8
CMPSSCTLEN		RESERVED					
R/W-0h		R-0-0h					
7	6	5	4	3	2	1	0
RESERVED							CMP1LDACOUTEN
R-0-0h							R/W-0h

**Table 10-19. CMPSSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CMPSSCTLEN	R/W	0h	0 - Rest of the configurations in this register are disabled 1 - Rest of the configuration in this register are enabled This bit is added for safety purpose. The configurations in this register are donot care if this bit is '0' Reset type: SYSRSn
14-1	RESERVED	R-0	0h	Reserved
0	CMP1LDACOUTEN	R/W	0h	0 - CMPSS1.COMPL is enabled and associated DAC will act as reference for comparator. 1 - CMPSS1.COMPL is disabled. Associated DAC will act as a general purpose DAC Reset type: SYSRSn

### 10.6.2.13 LOCK Register (Offset = 8Eh) [Reset = 0000000h]

LOCK is shown in [Figure 10-15](#) and described in [Table 10-20](#).

Return to the [Summary Table](#).

Lock Register

**Figure 10-15. LOCK Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
RESERVED					CMPSSCTL	VREGCTL	CMPLNMXSEL
R-0-0h					R/WOnce-0h	R/WOnce-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
CMPHNMXSEL	CMPLPMXSEL	CMPHPMXSEL	RESERVED	RESERVED	VMONCTL	ANAREFCTL	TSNSCTL
R/WOnce-0h							

**Table 10-20. LOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R-0	0h	Reserved
10	CMPSSCTL	R/WOnce	0h	CMPSSCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
9	VREGCTL	R/WOnce	0h	VREGCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
8	CMPLNMXSEL	R/WOnce	0h	CMPLNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
7	CMPHNMXSEL	R/WOnce	0h	CMPHNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
6	CMPLPMXSEL	R/WOnce	0h	CMPLPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
5	CMPHPMXSEL	R/WOnce	0h	CMPHPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
4	RESERVED	R/WOnce	0h	Reserved
3	RESERVED	R/WOnce	0h	Reserved
2	VMONCTL	R/WOnce	0h	VMONCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn

**Table 10-20. LOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	ANAREFCTL	R/WOnce	0h	ANAREFCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn
0	TSNSCTL	R/WOnce	0h	TSNSCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset. Reset type: SYSRSn

### 10.6.2.14 AGPIOTRLA Register (Offset = 10Ah) [Reset = 0000000h]

AGPIOTRLA is shown in [Figure 10-16](#) and described in [Table 10-21](#).

Return to the [Summary Table](#).

AGPIO Control Register

**Figure 10-16. AGPIOTRLA Register**

31	30	29	28	27	26	25	24
RESERVED	RESERVED	RESERVED	GPIO28	RESERVED	RESERVED	RESERVED	RESERVED
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	GPIO21	GPIO20	RESERVED	RESERVED	RESERVED	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	RESERVED	GPIO13	GPIO12	RESERVED	RESERVED	RESERVED	RESERVED
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

**Table 10-21. AGPIOTRLA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	GPIO28	R/W	0h	One time configuration for GPIO28 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	GPIO21	R/W	0h	One time configuration for GPIO21 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
20	GPIO20	R/W	0h	One time configuration for GPIO20 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved

**Table 10-21. AGPIOTRLA Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	GPIO13	R/W	0h	One time configuration for GPIO13 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
12	GPIO12	R/W	0h	One time configuration for GPIO12 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	RESERVED	R/W	0h	Reserved

### 10.6.2.15 AGPIOCTRLH Register (Offset = 118h) [Reset = 0000000h]

AGPIOCTRLH is shown in [Figure 10-17](#) and described in [Table 10-22](#).

Return to the [Summary Table](#).

AGPIO Control Register

**Figure 10-17. AGPIOCTRLH Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	GPIO242	RESERVED	RESERVED
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	GPIO230	RESERVED	GPIO228	GPIO227	GPIO226	RESERVED	GPIO224
R/W-0h							

**Table 10-22. AGPIOCTRLH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30	RESERVED	R/W	0h	Reserved
29	RESERVED	R/W	0h	Reserved
28	RESERVED	R/W	0h	Reserved
27	RESERVED	R/W	0h	Reserved
26	RESERVED	R/W	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23	RESERVED	R/W	0h	Reserved
22	RESERVED	R/W	0h	Reserved
21	RESERVED	R/W	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	GPIO242	R/W	0h	One time configuration for GPIO242 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
17	RESERVED	R/W	0h	Reserved
16	RESERVED	R/W	0h	Reserved
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R/W	0h	Reserved
13	RESERVED	R/W	0h	Reserved
12	RESERVED	R/W	0h	Reserved
11	RESERVED	R/W	0h	Reserved
10	RESERVED	R/W	0h	Reserved

**Table 10-22. AGPICTRLH Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	GPIO230	R/W	0h	One time configuration for GPIO230 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
5	RESERVED	R/W	0h	Reserved
4	GPIO228	R/W	0h	One time configuration for GPIO228 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
3	GPIO227	R/W	0h	One time configuration for GPIO227 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
2	GPIO226	R/W	0h	One time configuration for GPIO226 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn
1	RESERVED	R/W	0h	Reserved
0	GPIO224	R/W	0h	One time configuration for GPIO224 to decide whether AGPIO functionality is enabled 0 - AGPIO functionality is disabled 1 - AGPIO functionality is enabled Reset type: XRSn

This page intentionally left blank.

## Chapter 11

# Analog-to-Digital Converter (ADC)



The analog-to-digital converter (ADC) module described in this chapter is a Type 5 ADC. See the [C2000 Real-Time Control Peripheral Reference Guide](#) 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.

11.1 Introduction.....	1004
11.2 ADC Configurability.....	1007
11.3 SOC Principle of Operation.....	1010
11.4 SOC Configuration Examples.....	1014
11.5 ADC Conversion Priority.....	1016
11.6 Burst Mode.....	1019
11.7 EOC and Interrupt Operation.....	1021
11.8 Post-Processing Blocks.....	1023
11.9 Opens/Shorts Detection Circuit (OSDETECT).....	1027
11.10 Power-Up Sequence.....	1029
11.11 ADC Calibration.....	1029
11.12 ADC Timings.....	1030
11.13 Additional Information.....	1033
11.14 Software.....	1041
11.15 ADC Registers.....	1045

## 11.1 Introduction

The ADC module is a 12-bit successive approximation (SAR) style ADC. The ADC is composed of a core and a wrapper. The core is composed of the analog circuits which include the channel select MUX, the sample-and-hold (S/H) circuit, the successive approximation circuits, voltage reference circuits, and other analog support circuits. The wrapper is composed of the digital circuits that configure and control the ADC. These circuits include the logic for programmable conversions, result registers, interfaces to analog circuits, interfaces to the peripheral buses, post-processing circuits, and interfaces to other on-chip modules.

Each ADC module consists of a single sample-and-hold (S/H) circuit. The ADC module is designed to be duplicated multiple times on the same chip, allowing simultaneous sampling or independent operation of multiple ADCs. The ADC wrapper is start-of-conversion (SOC) based (see [Section 11.3](#)).

### 11.1.1 ADC Related Collateral

#### Foundational Materials

- [ADC Input Circuit Evaluation for C2000 MCUs \(TINA-TI\) Application Report](#)
- [C2000 Academy - ADC](#)
- [PSpice for TI design and simulation tool](#)
- [Real-Time Control Reference Guide](#)
  - Refer to the ADC section
- [TI Precision Labs - ADCs](#)
- [TI Precision Labs: Driving the reference input on a SAR ADC \(Video\)](#)
- [TI Precision Labs: Introduction to analog-to-digital converters \(ADCs\) \(Video\)](#)
- [TI Precision Labs: SAR ADC input driver design \(Video\)](#)
- [TI e2e: Connecting VDDA to VREFHI](#)
- [TI e2e: Topologies for ADC Input Protection](#)
- [TI e2e: Why does the ADC Input Voltage drop with sampling?](#)
  - Sampling a high impedance voltage divider with ADC
- [Understanding Data Converters Application Report](#)

#### Getting Started Materials

- [ADC-PWM Synchronization Using ADC Interrupt](#)
  - NOTE: This is a non-TI (third party) site.
- [Analog-to-Digital Converter \(ADC\) Training for C2000 MCUs \(Video\)](#)
- [Hardware Design Guide for F2800x C2000 Real-Time MCU Series](#)

#### Expert Materials

- [Analog Engineer's Calculator](#)
- [Analog Engineer's Pocket Reference](#)
- [Charge-Sharing Driving Circuits for C2000 ADCs \(using PSpice-FOR-TI\) Application Report](#)
- [Charge-Sharing Driving Circuits for C2000 ADCs \(using TINA-TI\) Application Report](#)
- [Debugging an integrated ADC in a microcontroller using an oscilloscope](#)
- [Methods for Mitigating ADC Memory Cross-Talk Application Report](#)
- [TI Precision Labs: ADC AC specifications \(Video\)](#)
- [TI Precision Labs: ADC Error sources \(Video\)](#)
- [TI Precision Labs: ADC Noise \(Video\)](#)
- [TI Precision Labs: Analog-to-digital converter \(ADC\) drive topologies \(Video\)](#)
- [TI Precision Labs: Electrical overstress on data converters \(Video\)](#)
- [TI Precision Labs: High-speed ADC fundamentals \(Video\)](#)
- [TI Precision Labs: SAR & Delta-Sigma: Understanding the Difference \(Video\)](#)
- [TI e2e: ADC Bandwidth Clarification](#)
- [TI e2e: ADC Calibration and Total Unadjusted Error](#)
- [TI e2e: ADC Reference Driver Options](#)

- [TI e2e: ADC Resolution with Oversampling](#)
- [TI e2e: ADC configuration for interleaved mode](#)
- [TI e2e: Simultaneous Sampling with Single ADC](#)

### 11.1.2 Features

Each ADC has the following features:

- 12-bit resolution
- Ratiometric external reference set by VREFHI and VREFLO pins
- Selectable internal reference of 2.5V or 3.3V
- Single-ended signal conversions
- Input multiplexer with up to 21 channels
- 16 configurable SOCs
- 16 individually addressable result registers
- Multiple trigger sources
  - S/W - software immediate start
  - All ePWMs - ADCSOC A or B
  - GPIO XINT2
  - CPU Timers 0/1/2
  - ADCINT1/2
- Four flexible PIE interrupts
- Configurable interrupt placement
- Burst mode
- Four post-processing blocks, each with:
  - Saturating offset calibration
  - Error from set-point calculation
  - High, low, and zero-crossing compare, with interrupt and ePWM trip capability
  - Trigger-to-sample delay capture

---

#### Note

Not every channel is pinned out from all ADCs. Check the device data sheet to determine which channels are available.

---

### 11.1.3 Block Diagram

Figure 11-1 shows the block diagram for the ADC core and ADC wrapper.

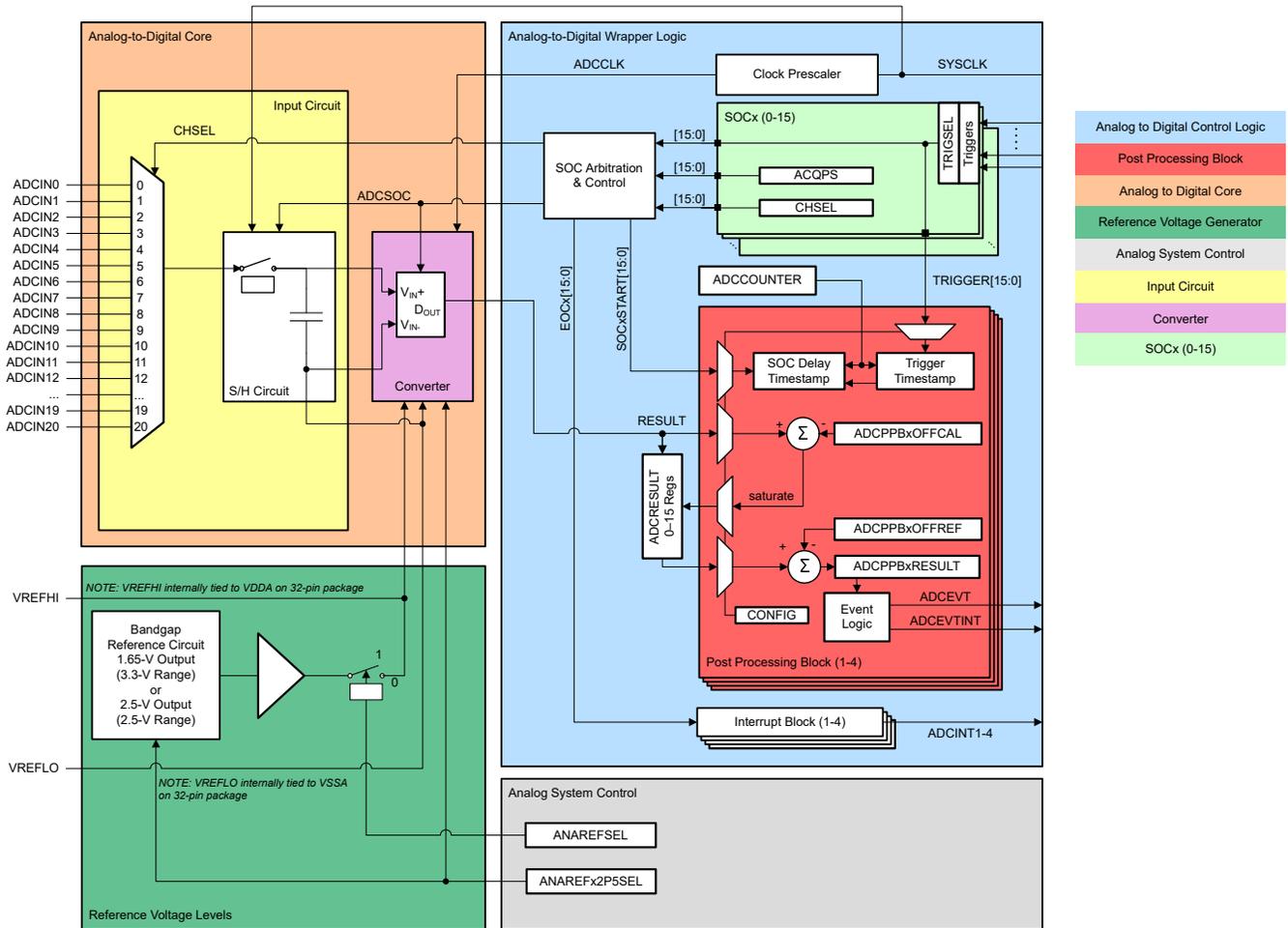


Figure 11-1. ADC Module Block Diagram

**Note**

The ADC block diagram reflects the number of ADC channels internally configurable on the device. The actual number of available external ADC inputs varies depending on part and package.

## 11.2 ADC Configurability

Some ADC configurations are individually controlled by the SOCs, while others are globally controlled per ADC module. [Table 11-1](#) summarizes the basic ADC options and the level of configurability. The subsequent sections discuss these configurations.

**Table 11-1. ADC Options and Configuration Levels**

Options	Configurability
Clock	Per module <sup>(1)</sup>
Resolution	Not configurable (12-bit only)
Signal mode	Not configurable (single-ended only)
Reference voltage source	Either external or internal for all modules
Trigger source	Per SOC <sup>(1)</sup>
Converted channel	Per SOC
Acquisition window duration	Per SOC <sup>(1)</sup>
EOC location	Per module
Burst Mode	Per module <sup>(1)</sup>

(1) Writing these values differently to different ADC modules can cause the ADCs to operate asynchronously. See [Section 11.13.1](#) for guidance on when the ADCs are operating synchronously or asynchronously.

### 11.2.1 Clock Configuration

The base ADC clock is provided directly by the system clock (SYSCLK). SYSCLK is used to generate the ADC acquisition window. The register ADCCTL2 has a PRESCALE field that determines the ADCCLK. ADCCLK is used to clock the converter, and is only active during the conversion phase. At all other times, including during the sample-and-hold window, the ADCCLK signal is gated off.

The core requires approximately 10.5 ADCCLK cycles to process a voltage into a conversion result. The user must determine the required duration of the acquisition window, see [Section 11.13.2](#).

#### Note

To determine an appropriate value for ADCCTL2.PRESCALE, see the device data sheet to determine the maximum SYSCLK and ADCCLK frequency.

### 11.2.2 Resolution

The resolution of the ADC determines how finely the analog range is quantized into digital values. Each ADC module supports a fixed resolution of 12 bits.

### 11.2.3 Voltage Reference

#### 11.2.3.1 External Reference Mode

The ADC modules share VREFHI and VREFLO inputs. In external reference mode, these pins are used as a ratiometric reference to determine the ADC conversion input range.

See [Section 11.13.6](#) for information on how to supply the reference voltage.

---

#### Note

- On devices with no external VREFHI and VREFLO pins, VREFHI is internally tied to VDDA and VREFLO is internally connected to the device analog ground, VSSA. Internal reference mode is not available for packages lacking VREFHI and VREFLO pins. See the device data sheet for packages with VREFHI and VREFLO pins available.
  - See the device data sheet to determine the allowable voltage range for VREFHI and VREFLO.
  - The external reference mode requires an external capacitor on the VREFHI pin. See the device data sheet for the specific value required.
- 

#### 11.2.3.2 Internal Reference Mode

In internal reference mode, the device drives a voltage out onto the VREFHI pin. The VREFHI and VREFLO pins then set the ADC conversion range. Internal reference mode is not available for packages without VREFHI and VREFLO pins. See the device data sheet for packages with VREFHI and VREFLO pins available.

The internal reference voltage can be configured to be either 2.5V or 1.65V. When the 1.65V internal reference voltage is selected, the ADC input signal is internally divided by 2 before conversion, which effectively makes the ADC conversion range from VREFLO to 3.3V.

---

#### Note

The internal reference mode also requires an external capacitor on the VREFHI pin. See the device data sheet for the specific value required.

---

#### 11.2.3.3 Selecting Reference Mode

The voltage reference mode must be configured by using either the ADC\_setVREF() or the SetVREF() functions, depending on the header files used, provided in C2000Ware. Using either of these functions makes sure that the correct trim is loaded in the ADC trim registers. This function must be called at least once after a device reset. Do not configure the voltage reference mode by directly writing to the ANAREFCTL register.

#### 11.2.4 Signal Mode

The ADC supports single-ended signaling.

In single-ended mode, the input voltage to the converter is sampled through a single pin (ADCINx), referenced to VREFLO.

### 11.2.5 Expected Conversion Results

Based on a given analog input voltage, the expected digital conversion is given in [Table 11-2](#). Fractional values are truncated.

**Table 11-2. Analog to 12-bit Digital Formulas**

Analog Input	Digital Result
when $ADCIN_y \leq VREFLO$	$ADCRESULT_x = 0$
when $VREFLO < ADCIN_y < VREFHI$	$ADCRESULT_x = 4096 \left( \frac{ADCIN_y - VREFLO}{VREFHI - VREFLO} \right)$
when $ADCIN_y \geq VREFHI$	$ADCRESULT_x = 4095$

### 11.2.6 Interpreting Conversion Results

Based on a given ADC conversion result, the corresponding analog input is given in [Table 11-3](#). This corresponds to the center of the possible range of analog voltages that can produce this conversion result.

**Table 11-3. 12-Bit Digital-to-Analog Formulas**

Digital Value	Analog Equivalent
when $ADCRESULT_y = 0$	$ADCIN_x \leq VREFLO$ (2)
when $0 < ADCRESULT_y < 4095$	$ADCIN_x = (VREFHI - VREFLO) \left( \frac{ADCRESULT_y}{4096} \right) + VREFLO$ (3)
when $ADCRESULT_y = 4095$	$ADCIN_x \geq VREFHI$ (4)

### 11.3 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 makes sure that the specified channel is captured using the specified acquisition window duration.

Multiple SOCs can be configured for the same trigger, channel, and acquisition window as desired. Configuring multiple SOCs to use the same trigger allows the trigger to generate a sequence of conversions. Configuring multiple SOCs to use the same trigger and channel allows for oversampling.

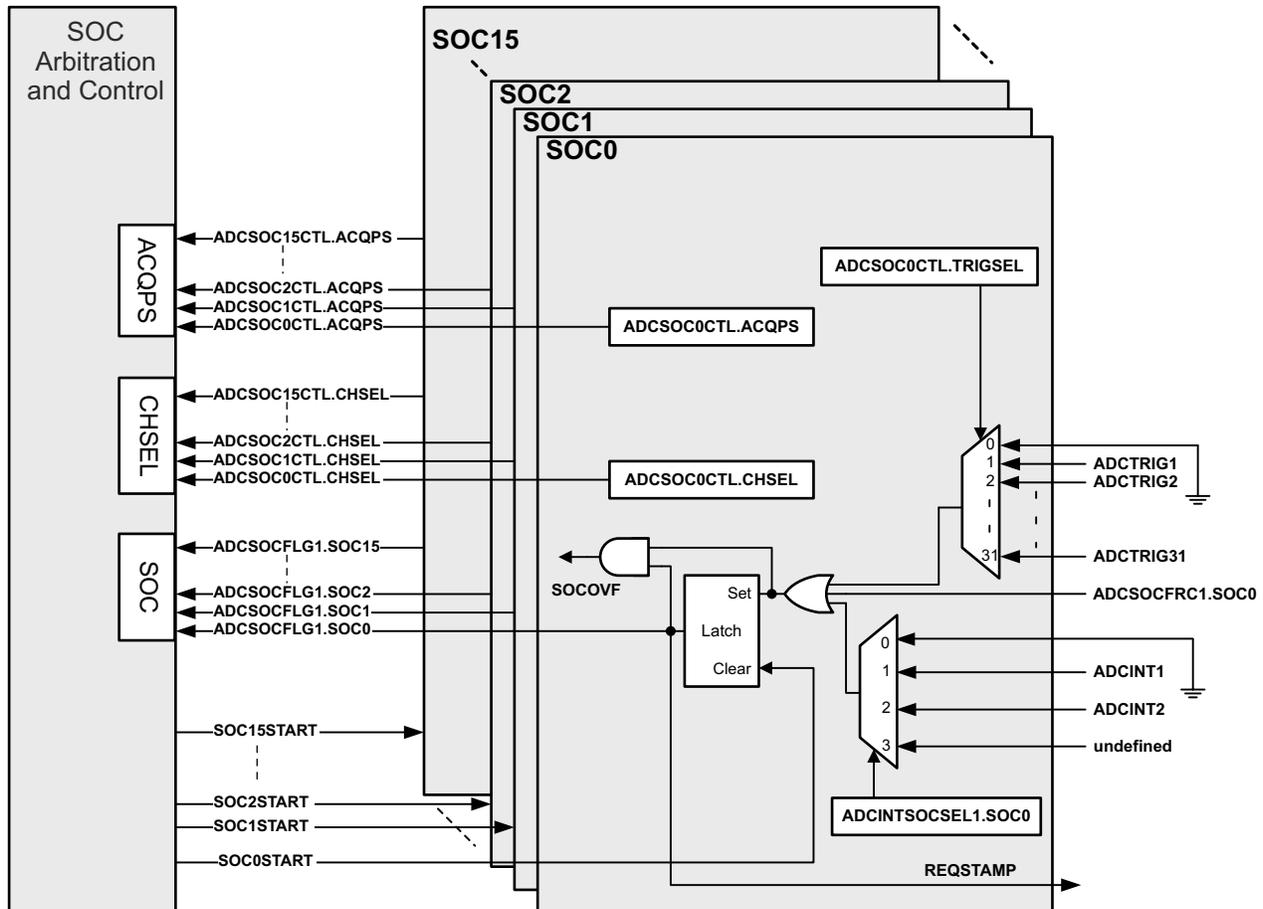


Figure 11-2. SOC Block Diagram

### 11.3.1 SOC Configuration

Each SOC has a configuration register, ADCSOCxCTL. Within this register, SOCx can be configured for trigger source, channel to convert, and acquisition (sample) window duration.

### 11.3.2 Trigger Operation

Each SOC can be configured to start on one of many input triggers. The primary trigger select for SOCx is in the ADCSOCxCTL.TRIGSEL register, which can select between:

- Disabled (software only)
- CPU Timers 0/1/2
- GPIO: Input X-Bar INPUT5
- ADCSOCA or ADCSOCB from each ePWM module

In addition, each SOC can also be triggered when the ADCINT1 flag or ADCINT2 flag is set. This is achieved by configuring the ADCINTSOCSEL1 register (for SOC0 to SOC7) or the ADCINTSOCSEL2 register (for SOC8 to SOC15). This is useful for creating continuous conversions.

**Table 11-4. ADC SOC Trigger Selection**

ADCSOCCTL.BURSTTRIGSEL	Signal
0	ADC_SOFTWARE_TRIGGER
1	CPU1_TINT0
2	CPU1_TINT1
3	CPU1_TINT2
4	INPUTXBAR5
5	EPWM1_ADCSOCA
6	EPWM1_ADCSOCB
7	EPWM2_ADCSOCA
8	EPWM2_ADCSOCB
9	EPWM3_ADCSOCA
10	EPWM3_ADCSOCB
11	EPWM4_ADCSOCA
12	EPWM4_ADCSOCB
13	EPWM5_ADCSOCA
14	EPWM5_ADCSOCB
15	EPWM6_ADCSOCA
16	EPWM6_ADCSOCB
17	EPWM7_ADCSOCA
18	EPWM7_ADCSOCB
19-31	Reserved

### 11.3.3 ADC Acquisition (Sample and Hold) Window

External signal sources vary in the ability to drive an analog signal quickly and effectively. To achieve rated resolution, the signal source needs to charge the sampling capacitor in the ADC core to within 0.5 LSBs 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.

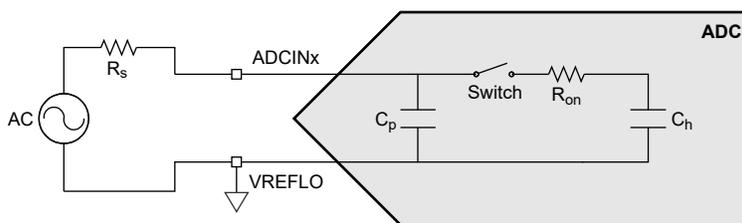
ACQPS is a 9-bit register that can be set to a value between 0 and 511, resulting in an acquisition window duration of:

$$\text{Acquisition window} = (\text{ACQPS} + 1) \cdot (\text{System Clock (SYSCLK) cycle time})$$

- 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 data sheet specifies a minimum acquisition window duration (in nanoseconds). The user is responsible for selecting an acquisition window duration that meets this requirement.

### 11.3.4 ADC Input Models

For single-ended operation, the ADC input characteristics for values in the single-ended input model (see [Figure 11-3](#)) can be found in the device data sheet.



**Figure 11-3. Single-Ended Input Model**

These input models must be used along with actual signal source impedance to determine the acquisition window duration. See [Section 11.13.2](#) for more information.

### 11.3.5 Channel Selection

Each SOC can be configured to convert any of the ADC channels. This behavior is selected for SOCx by the ADCSOCxCTL.CHSEL register. This is summarized in [Table 11-5](#).

**Table 11-5. 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
	16	ADCIN16
	17	ADCIN17
	18	ADCIN18
	19	ADCIN19
	20	ADCIN20

## 11.4 SOC Configuration Examples

The following sections provide some specific examples of how to configure the SOC5s to produce some conversions.

### 11.4.1 Single Conversion from ePWM Trigger

To configure ADC5 to perform a single conversion on channel ADCINA1 when the ePWM timer reaches the period match, a few things are necessary. First, ePWM3 must be configured to generate an SOCA or SOCB signal (in this statement, SOC refers to a signal in the ePWM module). See the *Enhanced Pulse Width Modulator Module (ePWM)* chapter on how to do this. Assume that SOCB was chosen.

SOC5 is chosen arbitrarily. Any of the 16 SOC5s can be used.

Assuming a 100ns sample window is desired with a SYSCLK frequency of 120MHz, then the acquisition window duration must be  $100\text{ns}/8.333\text{ns} = 12$  cycles. The ACQPS field must be set to  $12 - 1 = 11$ .

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 converts ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 11;    //SOC5 uses a sample duration of 12 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;  //SOC5 begins conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches the period and generates the SOCB signal, the ADC begins sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 begins sampling when SOC5 gains priority (see [Section 11.5](#)). The ADC control logic samples ADCINA1 with the specified acquisition window width of 100ns. Immediately after the acquisition is complete, the ADC begins converting the sampled voltage to a digital value. When the ADC conversion is complete, the results are available in the ADCRESULT5 register (see [Section 11.12](#) for exact sample, conversion, and result latch timings).

### 11.4.2 Oversampled Conversion from ePWM Trigger

To configure the ADC to oversample ADCINA1 4 times, we use the same configurations as the previous example, but apply them to SOC5, SOC6, SOC7, and SOC8.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 converts ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 9;      //SOC5 uses a sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;   //SOC5 begins conversion on ePWM3 SOCB
AdcaRegs.ADCSOC6CTL.bit.CHSEL = 1;      //SOC6 converts ADCINA1
AdcaRegs.ADCSOC6CTL.bit.ACQPS = 9;      //SOC6 uses a sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC6CTL.bit.TRIGSEL = 10;   //SOC6 begins conversion on ePWM3 SOCB
AdcaRegs.ADCSOC7CTL.bit.CHSEL = 1;      //SOC7 converts ADCINA1
AdcaRegs.ADCSOC7CTL.bit.ACQPS = 9;      //SOC7 uses a sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC7CTL.bit.TRIGSEL = 10;   //SOC7 begins conversion on ePWM3 SOCB
AdcaRegs.ADCSOC8CTL.bit.CHSEL = 1;      //SOC8 converts ADCINA1
AdcaRegs.ADCSOC8CTL.bit.ACQPS = 9;      //SOC8 uses a sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC8CTL.bit.TRIGSEL = 10;   //SOC8 begins conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches the period and generates the SOCB signal, the ADC begins sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 begins sampling when SOC5 gains priority (see [Section 11.5](#)). Once the conversion is complete for SOC5, SOC6 begins converting ADCINA1 and the results for SOC5 are placed in the ADCRESULT5 register. All four conversions eventually are 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 can begin converting SOC6, SOC7, or SOC8 before SOC5 depending on the position of the round-robin pointer when the ePWM trigger is received. See [Section 11.5](#) to understand how the next SOC to be converted is chosen.

### 11.4.3 Multiple Conversions from CPU Timer Trigger

This example shows how to sample multiple signals with different acquisition window requirements. CPU1 Timer 2 is 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 the 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 11-6](#), where a SYCLK of 120MHz is assumed (8.333ns cycle time).

**Table 11-6. Example Requirements for Multiple Signal Sampling**

Signal Name	Acquisition Window Requirement	Acquisition Window (SYSCLK Cycles)	ACQPS Register Value
Signal 1	>200ns	200ns/8.333ns = 24	24 – 1 = 23
Signal 2	>740ns	740ns/8.333ns = 89 (round up)	89 – 1 = 88
Signal 3	>183.33ns	183.33ns/8.333ns = 22	22 – 1 = 21
Signal 4	>485ns	485ns/8.333ns = 59 (round up)	59 – 1 = 58

Next decide which ADC pins to connect to each signal. This is highly dependent on the application board layout. Once the pins are selected, determining the value of CHSEL is straightforward (see [Table 11-7](#)).

**Table 11-7. 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, generate the SOC configurations:

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 5;           //SOC0 converts ADCINA5
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 23;         //SOC0 uses a sample duration of 24 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 3;        //SOC0 begins conversion on CPU1 Timer 2
AdcaRegs.ADCSOC1CTL.bit.CHSEL = 0;         //SOC1 converts ADCINA0
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 88;        //SOC1 uses a sample duration of 89 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 3;        //SOC1 begins conversion on CPU1 Timer 2
AdcaRegs.ADCSOC2CTL.bit.CHSEL = 3;         //SOC2 converts ADCINA3
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 21;        //SOC2 uses a sample duration of 22 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 3;        //SOC2 begins conversion on CPU1 Timer 2
AdcaRegs.ADCSOC3CTL.bit.CHSEL = 2;         //SOC3 converts ADCINA2
AdcaRegs.ADCSOC3CTL.bit.ACQPS = 58;        //SOC3 uses a sample duration of 59 SYSCLK cycles
AdcaRegs.ADCSOC3CTL.bit.TRIGSEL = 3;        //SOC3 begins conversion on CPU1 Timer 2

```

As configured, when CPU1 Timer 2 generates an event, SOC0, SOC1, SOC2, and SOC3 eventually is sampled and converted, in that order. The conversion results for ACINA5 (Signal 1) are in ADCRESULT0. Similarly, The results for ADCINA0 (Signal 2), ADCINA3 (Signal 3), and ADCINA2 (Signal 4) are in ADCRESULT1, ADCRESULT2, and ADCRESULT3, respectively.

#### Note

There is a possibility, but unlikely, that the ADC can 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 [Section 11.5](#) to understand how the next SOC to be converted is chosen.

#### 11.4.4 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 can be accomplished by the statement:

```
AdcaRegs.ADCSOCFRC1.a11 = 0x000F;           //set SOC flags for SOC0 to SOC3
```

#### 11.5 ADC Conversion Priority

When multiple SOC flags are set at the same time, one of two forms of priority determines the converted order. The default priority method is round-robin. In this scheme, no SOC has an inherent higher priority than another. Priority depends on the round-robin pointer (RRPOINTER). The RRPOINTER reflected in the ADCSOCPRIORITYCTL register points to the last SOC converted. The highest priority SOC is given to the next value greater than the RRPOINTER value, wrapping around back to SOC0 after SOC15. At reset the value is 16 since 0 indicates a conversion has already occurred. When RRPOINTER equals 16 the highest priority is given to SOC0. The RRPOINTER is reset when the ADC module is reset or when the reset value is written to the SOCPRIORITY register. The ADC module is reset by writing and clearing the SOFTPRES bit corresponding to the ADC instance.

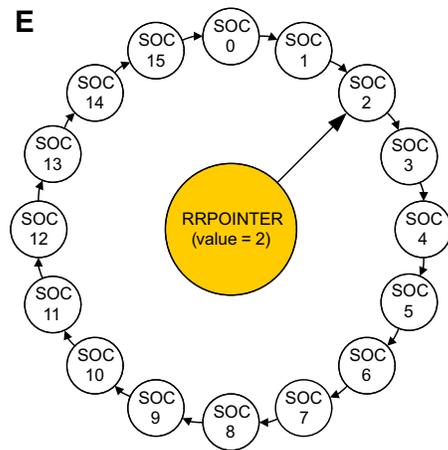
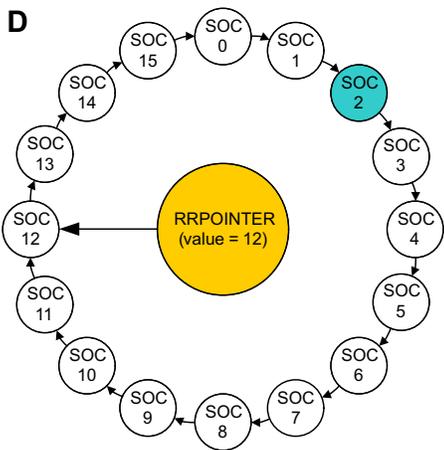
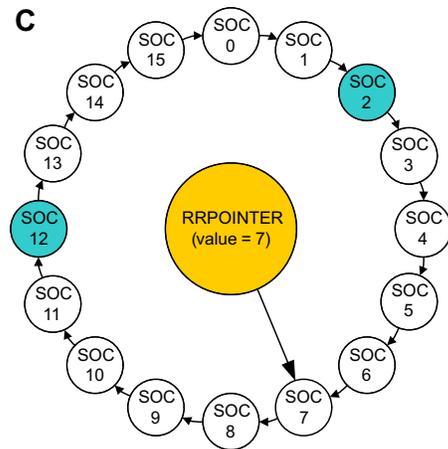
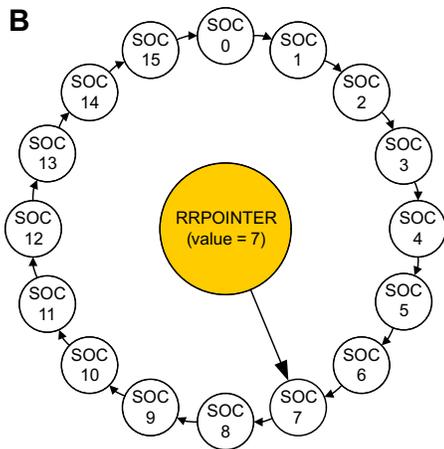
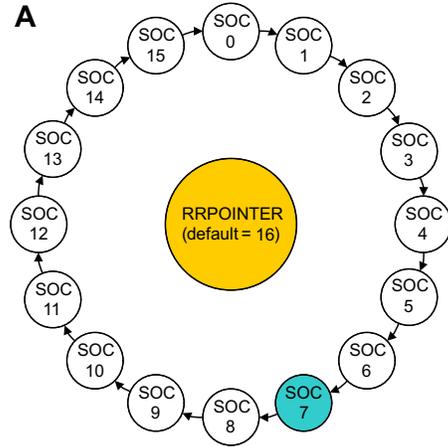
An example of the round-robin priority method is given in [Figure 11-4](#).

The SOCPRIORITY field in the ADCSOCPRIORITYCTL register can be used to assign high priority from a single to all of the SOC's. When configured as high priority, an SOC interrupts the round-robin wheel after any current conversion completes and inserts in as the next conversion. After the conversion completes, the round-robin wheel continues where the conversion was interrupted. If two high priority SOC's are triggered at the same time, the SOC with the lower number takes precedence.

High priority mode is assigned first to SOC0, then in increasing numerical order. The value written in the SOCPRIORITY field defines the first SOC that is not high priority. In other words, if a value of 4 is written into SOCPRIORITY, 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 11-5](#).

- 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 .



**Figure 11-4. Round Robin Priority Example**

Example when SOC PRIORITY = 4

- A** After reset, SOC4 is 1<sup>st</sup> on round robin wheel ;  
SOC7 receives trigger ;  
SOC7 configured channel is converted immediately .
- B** RRPOINTER changes to point to SOC 7 ;  
SOC8 is now 1<sup>st</sup> on round robin wheel .
- C** SOC2 & SOC12 triggers rcvd. simultaneously ;  
SOC2 interrupts round robin wheel and SOC 2 configured channel is converted while SOC 12 stays pending .
- D** RRPOINTER stays pointing to 7 ;  
SOC12 configured channel is now converted .
- E** RRPOINTER changes to point to SOC 12 ;  
SOC13 is now 1<sup>st</sup> on round robin wheel .

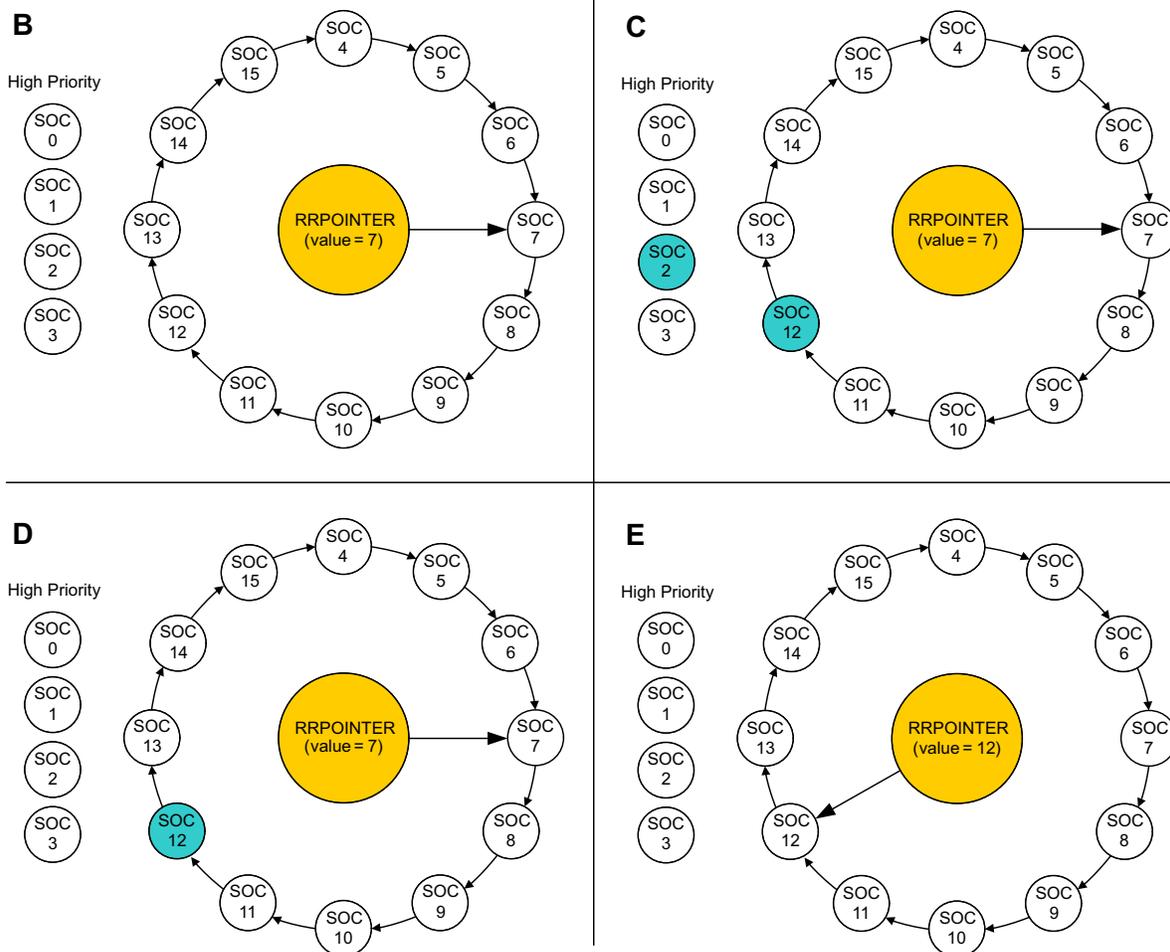


Figure 11-5. High Priority Example

## 11.6 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 does not set all round-robin SOC's to be converted, but only (ADCBURSTCTL.BURSTSIZE + 1) SOC's. The first SOC to be set is the SOC with the highest priority based on the round-robin pointer, and subsequent SOC's are 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. The value of (ADCBURSTCTL.BURSTSIZE + 1) must be less than or equal to the number of SOC's configured for round-robin priority.

For example, if SOCPRIORITY = 12, that is, SOC12, SOC13, SOC14, and SOC15 are in round-robin, ADCBURSTCTL.BURSTSIZE setting must be  $\leq 3$  for burst mode to operate correctly.

### 11.6.1 Burst Mode Example

Burst mode can be used to sample a different set of signals on every other trigger. In the following example, ADCIN7 and ADCIN5 are converted on the first trigger from CPU1 Timer 2 and every other trigger thereafter. ADCIN2 and ACIN3 are converted on the second trigger from CPU1 Timer 2 and every other trigger thereafter. All signals are converted with 20 SYSCLK cycle wide acquisition windows, but different durations can be configured for each SOC as desired.

```
AdcaRegs.BURSTCTL.BURSTEN = 1;           //Enable ADC burst mode
AdcaRegs.BURSTCTL.BURSTTRIG = 3;         //CPU1 Timer 2 triggers burst of conversions
AdcaRegs.BURSTCTL.BURSTSIZE = 1;        //conversion bursts are 1 + 1 = 2 conversions long
AdcaRegs.SOCPRICL.bit.SOCPRIORITY = 12; //SOC0 to SOC11 are high priority
AdcaRegs.ADCSOC12CTL.bit.CHSEL = 7;      //SOC12 converts ADCINA7
AdcaRegs.ADCSOC12CTL.bit.ACQPS = 19;     //SOC12 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC13CTL.bit.CHSEL = 5;      //SOC13 converts ADCINA5
AdcaRegs.ADCSOC13CTL.bit.ACQPS = 19;     //SOC13 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC14CTL.bit.CHSEL = 2;      //SOC14 converts ADCINA2
AdcaRegs.ADCSOC14CTL.bit.ACQPS = 19;     //SOC14 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC15CTL.bit.CHSEL = 3;      //SOC15 converts ADCINA3
AdcaRegs.ADCSOC15CTL.bit.ACQPS = 19;     //SOC15 uses sample duration of 20 SYSCLK cycles
```

When the first CPU1 Timer 2 trigger is received, SOC12 and SOC13 are converted immediately if the ADC is idle. If the ADC is busy, SOC12 and SOC13 are converted once the SOC's gain priority. The results for SOC12 and SOC13 are in ADCRESULT12 and ADCRESULT13, respectively. After SOC13 completes, the round-robin pointer gives the highest priority to SOC14. Because of this, when the next CPU1 Timer 2 trigger is received, SOC14 and SOC15 is set as pending and eventually converted. The results for SOC14 and SOC15 are in ADCRESULT14 and ADCRESULT15, respectively. Subsequent triggers continue to toggle between converting SOC12 and SOC13, and converting SOC14 and SOC15.

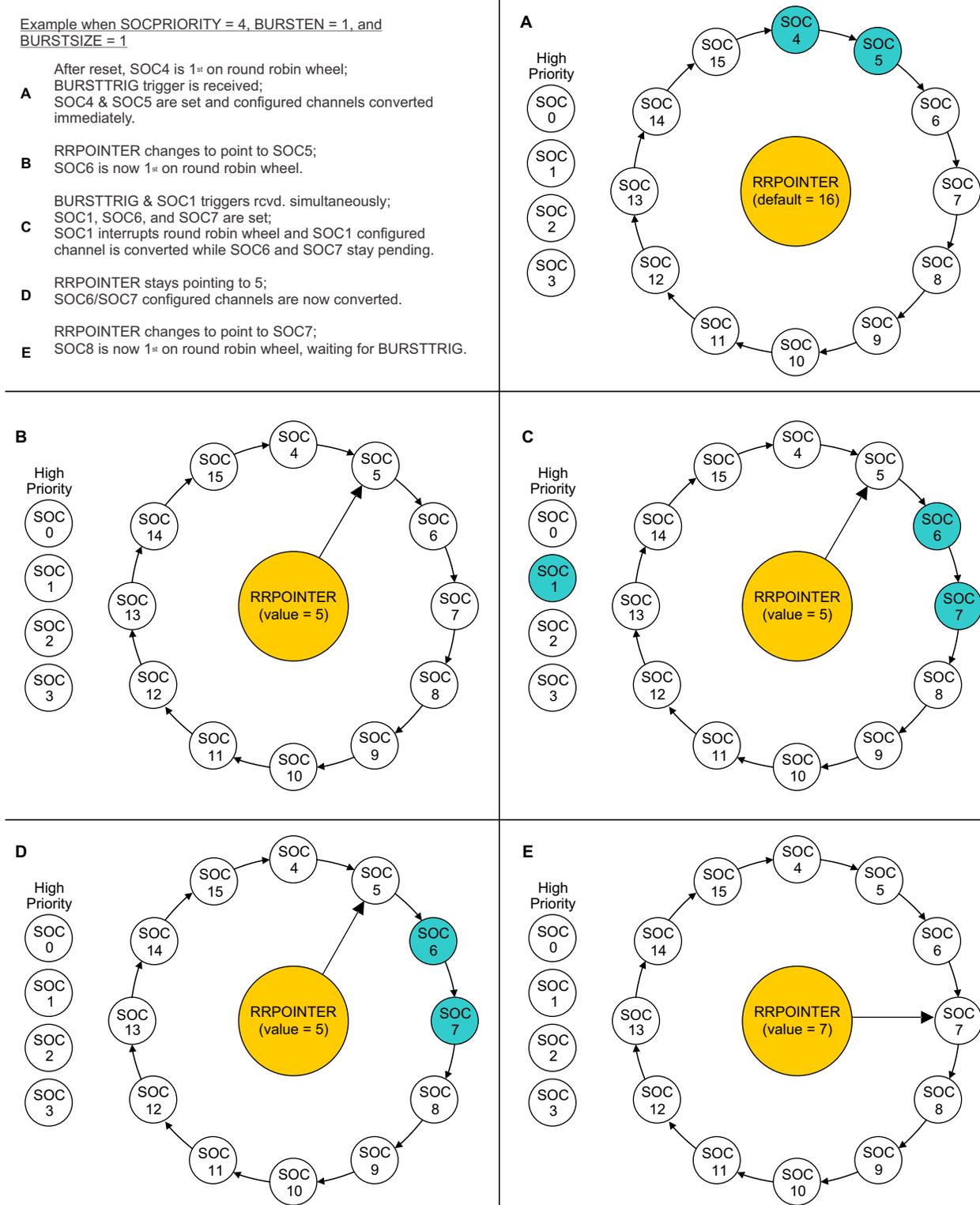
While the above example toggles between two sets of conversions, three or more different sets of conversions can be achieved using a similar approach.

### 11.6.2 Burst Mode Priority Example

An example of priority resolution using burst mode and high-priority SOC's is presented in [Figure 11-6](#).

Example when  $SOC_{PRIORITY} = 4$ ,  $BURSTEN = 1$ , and  $BURSTSIZE = 1$

- A** After reset, SOC4 is 1<sup>st</sup> on round robin wheel; BURSTTRIG trigger is received; SOC4 & SOC5 are set and configured channels converted immediately.
- B** RRPOINTER changes to point to SOC5; SOC6 is now 1<sup>st</sup> on round robin wheel.
- C** BURSTTRIG & SOC1 triggers rcvd. simultaneously; SOC1, SOC6, and SOC7 are set; SOC1 interrupts round robin wheel and SOC1 configured channel is converted while SOC6 and SOC7 stay pending.
- D** RRPOINTER stays pointing to 5; SOC6/SOC7 configured channels are now converted.
- E** RRPOINTER changes to point to SOC7; SOC8 is now 1<sup>st</sup> on round robin wheel, waiting for BURSTTRIG.



**Figure 11-6. Burst Priority Example**

### 11.7 EOC and Interrupt Operation

Each SOC has a corresponding end-of-conversion (EOC) signal. This EOC signal can be used to trigger an ADC interrupt. The ADC can be configured to generate the EOC pulse at either the end of the acquisition window or at the end of the voltage conversion. This is configured using the bit INTPULSEPOS in the ADCCTL1 register. See Section 11.12 for exact EOC pulse location.

Each ADC module has 4 configurable ADC interrupts. These interrupts can be triggered by any of the 16 EOC signals. The flag bit for each ADCINT can be read directly to determine if the associated SOC is complete or the interrupt can be passed on to the PIE.

**Note**

The ADCCTL1.ADCBSY bit being clear does not indicate that all conversions in a set of SOCs have completed, only that the ADC is ready to process the next conversion. To determine if a sequence of SOCs is complete, link an ADCINT flag to the last SOC in the sequence and monitor that ADCINT flag.

Figure 11-7 shows a block diagram of the ADC interrupt structure.

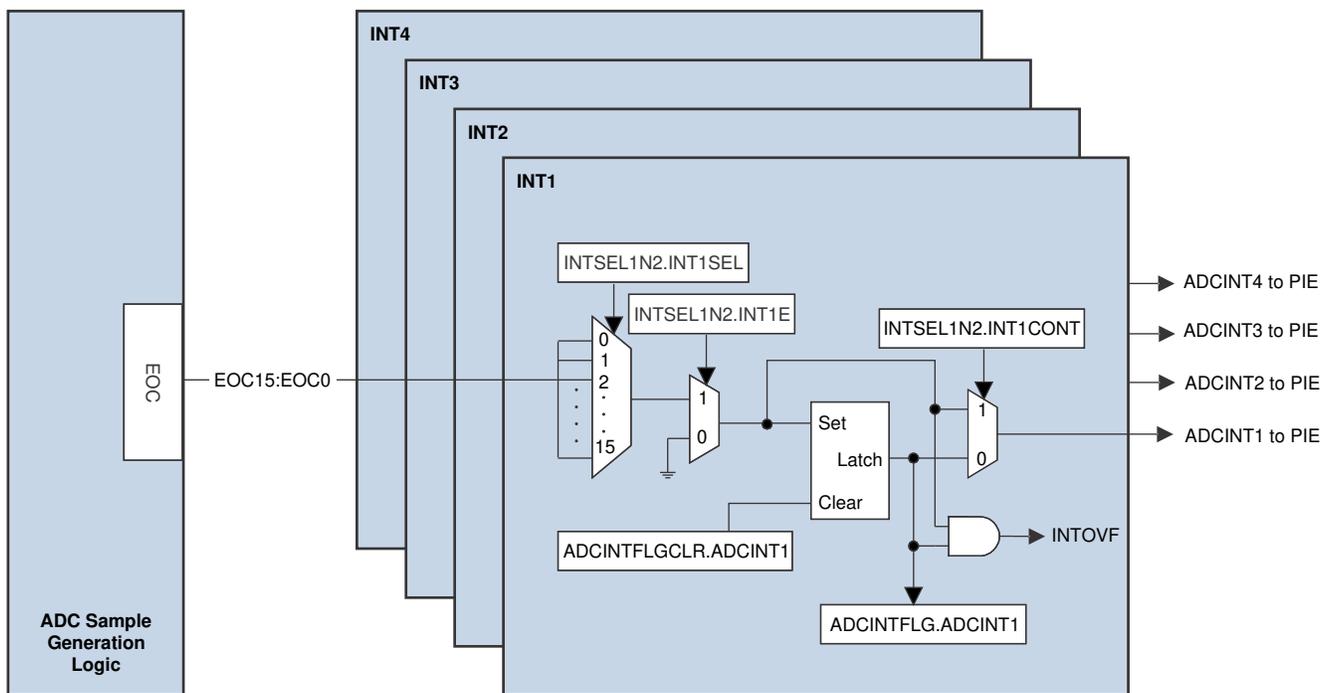


Figure 11-7. ADC EOC Interrupts

### 11.7.1 Interrupt Overflow

If the EOC signal sets a flag in the ADCINTFLG register, but that flag is already set, an interrupt overflow occurs. By default, overflow interrupts are not passed on to the PIE module. When an overflow occurs on a given flag in the ADCINTFLG register, the corresponding flag in the ADCINTOVF register is set. This overflow flag is only used to detect that an overflow has occurred; the flag does not block further interrupts from propagating to the PIE module.

When an ADC interrupt overflow occurs, the application must check the appropriate ADCINTOVF flag inside the ISR or in the background loop and take appropriate action when an overflow is detected. The following code snippets demonstrate how to check the ADCINTOVF flag inside the ISR after attempting to clear the ADCINT flag.

```
// Clear the interrupt flag
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;    //clear INT1 flag for ADC-A

// check if an overflow has occurred
if(1 == AdcaRegs.ADCINTOVF.bit.ADCINT1)    //ADCINT overflow occurred
{
    AdcaRegs.ADCINTOVFCLR.bit.ADCINT1 = 1    //clear overflow flag
    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1    //Re-clear ADCINT flag
}
```

```
//
// Clear the interrupt flag
//
ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);

//
// Check if an overflow has occurred
//
if(true == ADC_getInterruptOverflowStatus(ADCA_BASE, ADC_INT_NUMBER1))
{
    ADC_clearInterruptOverflowStatus(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
}
```

### 11.7.2 Continue to Interrupt Mode

The INTxCONT bits in the ADCINTSEL1N2 and ADCINTSEL3N4 registers configure how interrupts are handled when an ADCINTFLG has not yet been cleared from a prior interrupt. This mode is disabled by default and additional overlapping interrupts are not issued to the PIE. By activating this mode, ADC interrupts always reach the PIE. If interrupts occur while ADCINTFLG is set, the ADCINTOVF register remains set regardless of the configuration of the INTxCONT bits.

### 11.7.3 Early Interrupt Configuration Mode

Enabling early interrupt mode can allow the application to enter the ADC interrupt service routine before the ADC results are ready. This allows the application to do any necessary pre-work so that the application can act on the ADC results immediately when the ADC results become available. If the timing of the early interrupt is too early, then the application needs to waste time until the updated ADC results become available. To prevent this situation, the time the ADC interrupt is entered in early interrupt mode is configurable by way of the DELAY field in the ADCINTCYCLE register.

- To use the configurable interrupt time, the ADC must be in early interrupt mode. To achieve this, clear the bit INTPULSEPOS to 0 in ADCCTL1.
- The DELAY value in the ADCINTCYCLE register sets the number of additional SYSCLK cycles after the falling edge of the SOC pulse before the ADCINT flag is set.
- If the value of DELAY goes beyond EOC, the ADC interrupt is generated along with EOC.
- Writing values to DELAY when INTPULSEPOS is set to 1 does not have any effect on the interrupt generation.

### 11.8 Post-Processing Blocks

Each ADC module contains four post-processing blocks (PPB). These blocks can be associated with any of the 16 RESULT registers using the ADCPPBxCONFIG.CONFIG bit field. The post-processing blocks have the ability to:

- Remove an offset associated with the ADCIN channel
- Subtract out a reference value
- Flag a zero-crossing point, with the option to trip a PWM and generate an interrupt
- Flag a high or low compare limit, with the option to trip a PWM and generate an interrupt
- Record the delay between the associated SOC trigger and when sampling actually begins

Figure 11-8 presents the structure of each PPB. Subsequent sections explain the use of each submodule.

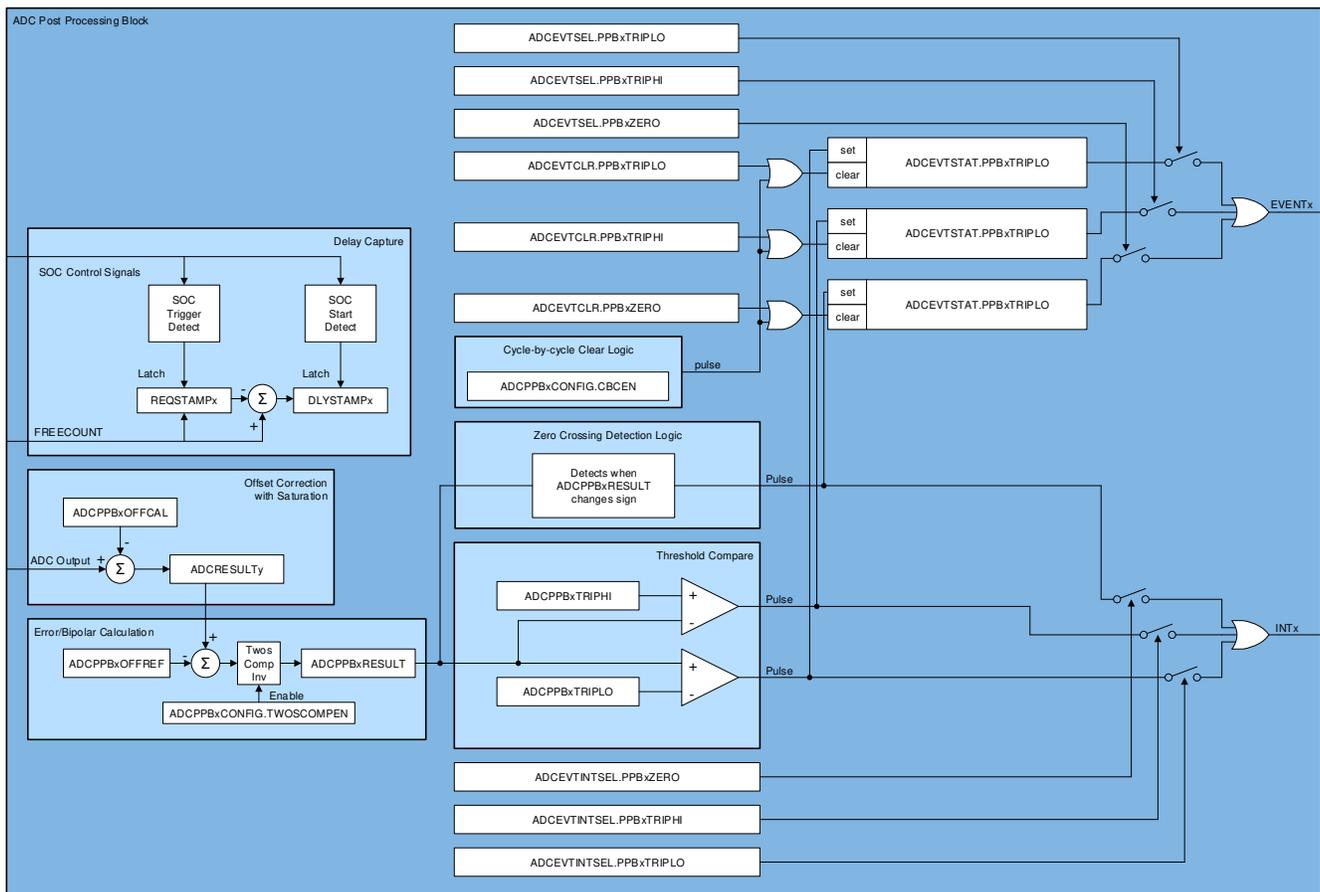


Figure 11-8. ADC PPB Block Diagram

### 11.8.1 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 automatically adds or subtracts the value in the OFFCAL register from the raw conversion result and stores the value in the ADCRESULT register. This addition/subtraction saturates at 0 on the low end and 4095 on the high end.

---

#### Note

- Writing a 0 to the OFFCAL register effectively disables the offset correction feature, passing the raw result unchanged to the ADCRESULT register.
  - To point multiple PPBs to the same SOC is possible. In this case, the OFFCAL value that is actually applied comes from the PPB with the highest number.
  - In particular, care needs to be taken when using the PPB on SOC0, as all PPBs point to this SOC by default. This can cause unintentional overwriting of offset correction of a lower numbered PPB by a higher numbered PPB.
- 

### 11.8.2 PPB Error Calculation

In many applications, an error from a set point 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 functions 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 automatically subtracts the value in the OFFREF register from the ADCRESULT value and stores the value in the ADCPPBxRESULT register. This subtraction produces 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.

---

#### Note

- Do not write a value larger than 12 bits to the ADCPPBxOFFREF register.
  - Since the ADCPPBxRESULT register is unique for each PPB, to point multiple PPBs to the same SOC and get different results for each PPB is possible.
  - Writing a 0 to the ADCPPBxOFFREF register effectively disables the error calculation feature, passing the ADCRESULT value unchanged to the ADCPPBxRESULT register.
  - Writing a new value to ADCPPBxOFFREF causes an immediate update to the ADCPPBxRESULT register. However, the flags coming out of the PPB do not change until the next end-of-conversion (EOC). For instance, if changing the ADCPPBxOFFREF register causes ADCPPBxRESULT to change signs, but the next conversion brings the result back to the same sign as before the OFFREF change, no ADCPPBxZERO flag is set.
- 

### 11.8.3 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 high and low limits, or whenever ADCPPBxRESULT changes sign. Based on these comparisons, the PPB can generate a trip to the PWM and an interrupt automatically, lowering the sample to

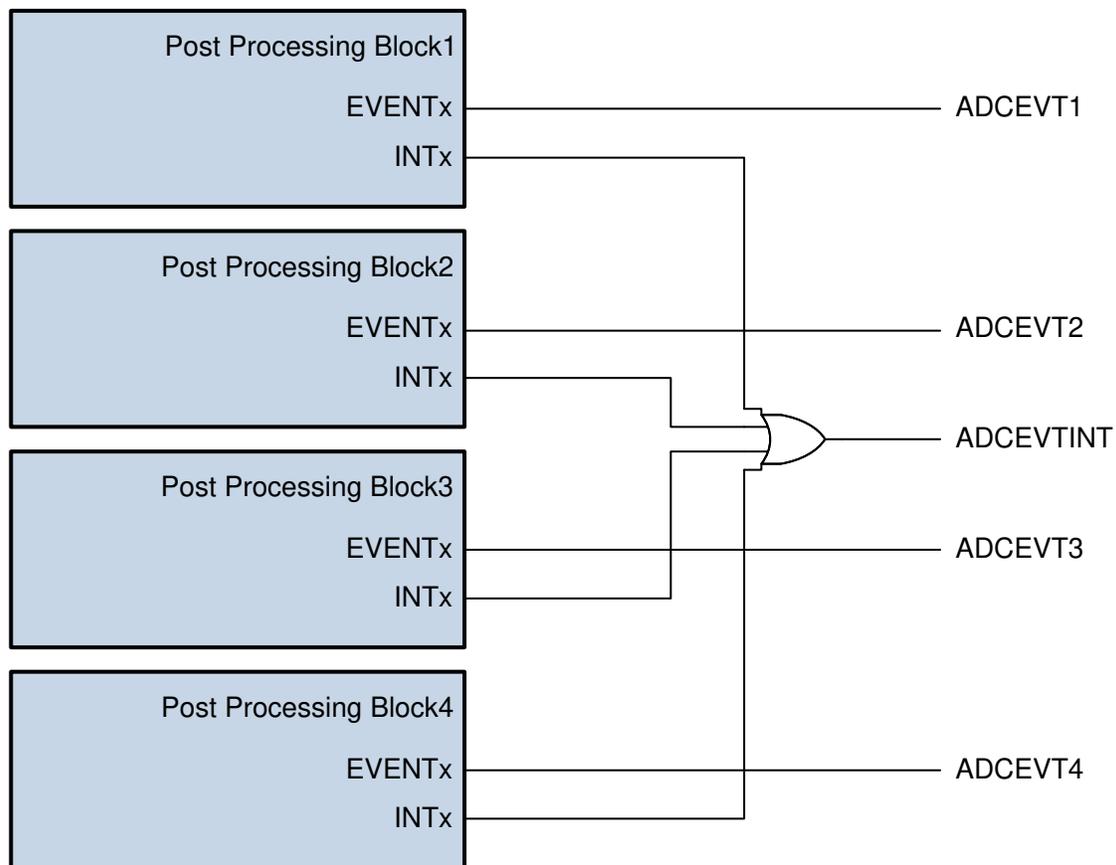
ePWM latency and reducing software overhead. This functionality also enables safety-conscious applications to trip the ePWM based on an out-of-range ADC conversion without any CPU intervention.

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 is set in the ADCEVTSTAT register. Note that the PPBxZERO bit in the ADCEVTSTAT register is gated by end-of-conversion (EOC), not by the sign change in the ADCPPBxRESULT register. The ADCEVTCLR register has corresponding bits to clear these event flags. The ADCEVTSEL register has corresponding bits which allow the events to propagate through to the PWM. The ADCEVTINTSEL register has corresponding bits that 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 11-9](#).

**Note**

- If different actions need to be taken for different PPB events from the same ADC module, then the ADCEVTINT ISR has 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 zero-crossing, this can be achieved by pointing multiple PPBs to the same SOC.
- The zero-crossing detect circuit considers a result of zero to be positive.



**Figure 11-9. ADC PPB Interrupt Event**

### 11.8.4 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 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 contains the number of SYSCLK cycles between when the associate SOC was triggered and when the SOC 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 can overflow more than once, leading to incorrect DLYSTAMP value. Be cautious when using very slow conversions to prevent this from happening.

The sample delay capture does not function, if the associated SOC is triggered using software. The sample delay capture, however, correctly records the delay, if the software triggering of a different SOC causes the SOC associated with the PPB to be delayed

---

### 11.9 Opens/Shorts Detection Circuit (OSDETECT)

The opens/shorts detection circuit (OSDETECT) can be used to detect pin faults in the system. The circuit connects to the ADC input after the channel select multiplexer but before the S+H circuit as shown in Figure 11-10.

**Note**

- The divider resistance tolerances can vary widely; hence, this feature must not be used to check for conversion accuracy.
- See the data sheet for implementation and availability of analog input channels.
- Due to high drive impedance, a S+H duration much longer than the ADC minimum is needed.

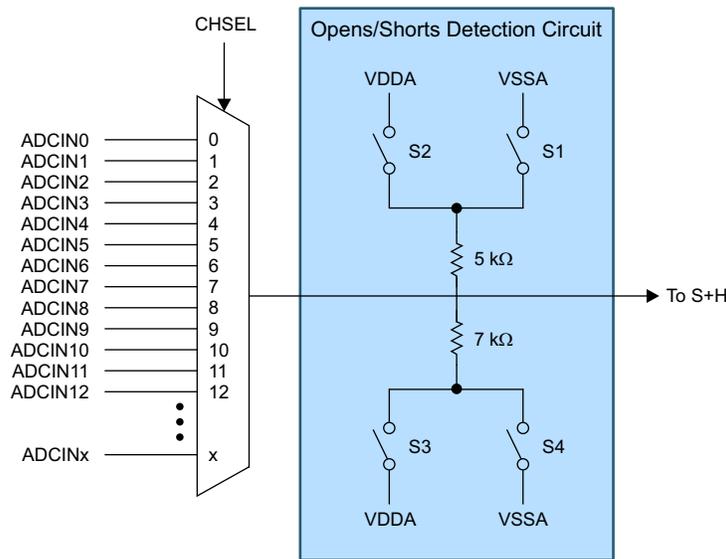


Figure 11-10. Opens/Shorts Detection Circuit

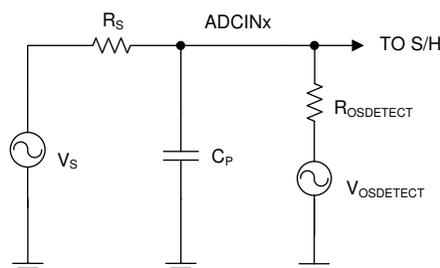
The circuit can be operated by writing a value to the DETECTCFG field in the ADCOSDETECT register. This causes the circuit to source a voltage onto the input during the S+H phase of any conversion. The voltage and drive strength of the OSDETECT circuit for different DETECTCFG settings is given in Table 11-8.

Table 11-8. DETECTCFG Settings

ADCOSDETECT. DETECTCFG	Source Voltage	S4	S3	S2	S1	Drive Impedance
0	Off	Open	Open	Open	Open	Open
1	Zero Scale	Closed	Open	Open	Closed	5K    7K
2	Full Scale	Open	Closed	Closed	Open	5K    7K
3	5/12 VDDA	Open	Closed	Open	Closed	5K    7K
4	7/12 VDDA	Closed	Open	Closed	Open	5K    7K
5	Zero Scale	Open	Open	Open	Closed	5K
6	Full Scale	Open	Open	Closed	Open	5K
7	Zero Scale	Closed	Open	Open	Open	7K

### 11.9.1 Implementation

A representative circuit with the OSDETECT implementation consists of the signal source with series resistance  $R_S$ , shunt capacitor  $C_P$ , the equivalent OSDETECT resistance  $R_{OSDETECT}$  and voltage  $V_{OSDETECT}$  is shown in Figure 11-11 and can be used as a basis to calculate the signal level going in to the sampling capacitor.  $R_{OSDETECT}$  and  $V_{OSDETECT}$  are the equivalent input resistance and voltage source contributed by the OSDETECT circuit with values shown in Table 11-8 for the different configuration settings. Refer to Figure 11-11 when deriving the input signal to S/H if signal source  $V_S$  is driving while the OSDETECT feature is enabled.



**Figure 11-11. Input Circuit Equivalent with OSDETECT Enabled**

The input impedance  $R_S$  and  $C_P$  are integral parts of the signal source or can have been implemented in the design to precondition the signal or to control signal settling time to meet S/H requirements. The input path has to be considered when using the OSDETECT feature, as this affects the conversion results. For instance, driving an input signal when this feature is enabled connects signal  $V_S$  to the OSDETECT circuit through  $R_S$  and affects the ADC results. Larger  $C_P$  values (in the order greater than hundreds of pF) require using higher ACQPS to make sure the signal at the input has settled prior to conversion.

To enable the circuit:

1. Configure the ADC for conversion (for example, channel, SOC, ACQPS, prescaler, trigger, and so on).
2. Set up the ADCOSDETECT register for the desired voltage divider connection as shown in Table 11-8.
3. Initiate a conversion and inspect the conversion result.

Interpret the results based on what is driving on the input side and what are the values of  $R_S$  and  $C_P$ . If the  $V_S$  signal can be disconnected from the input pin, the circuit can be used to detect open and shorted input pins as described in the following sections.

#### 11.9.2 Detecting an Open Input Pin

By cycling through the various OSDETECT settings, the input signal is pulled towards the sourced voltages. An input with good drive strength (pin not open) is minimally affected. However, if the pin is open, the sampled voltages is close to the source voltages specified in Table 11-8.

#### 11.9.3 Detecting a Shorted Input Pin

By cycling through the various OSDETECT settings, the input signal is pulled towards the sourced voltages. An input with finite drive strength (pin not shorted) is pulled toward each sourced voltage. However, if the pin is shorted, the signal remains at the same voltage.

## 11.10 Power-Up Sequence

Upon device power-up or system level reset, the ADC is powered down and disabled. When powering up the ADC, use the following sequence:

1. Set the bit to enable the desired ADC clock in the PCLKCR13 register.
2. Set the desired ADC clock divider in the PRESCALE field of ADCCTL2.
3. Power up the ADC by setting the ADCPWDNZ bit in ADCCTL1.
4. Allow a delay before sampling. See the data sheet for the necessary time.

If multiple ADCs are powered up simultaneously, steps 1 and step 3 can each be done for all ADCs in one write instruction. Also, only one delay is necessary as long as the delay occurs after all the ADCs have begun powering up.

## 11.11 ADC Calibration

During the fabrication and test process, Texas Instruments calibrates the gain, offset, and linearity of the ADCs. These trim settings are stored in TI reserved OTP memory, and can be loaded using C-callable functions.

- The Device\_cal() function copies the trim values for ADC from OTP memory to the respective trim registers.

Until the appropriate factory trim is loaded, the ADC and other analog modules are not specified to operate within the data sheet specifications. Similarly, if trim values other than the factory settings are placed into the trim registers, the ADC (and other modules) is not specified to operate within the data sheet specifications.

The boot ROM calls the calibration functions, so trim values are 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 must call the calibration functions (defined in the C2000Ware header files).

### 11.11.1 ADC Zero Offset Calibration

ADC offset error is determined and calibrated during factory testing. However, the user still has the option to perform offset calibration if the end application specifically requires this. This section describes how to perform offset calibration using internal VREFLO connection for single-ended operation.

Zero offset error is defined as the difference from 0 that occurs when converting a voltage at VREFLO. The zero offset error can be positive or negative. To correct this error, an adjustment of equal magnitude and opposite polarity is written into the ADCOFFTRIM register. The value contained in this register is 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 is not affected, and the full dynamic range of the ADC is maintained for any trim value.

---

#### Note

Regardless of the converter resolution, the size of each ADCOFFTRIM step is (VREFHI-VREFLO)/65536.

---

Use the following procedure to re-calibrate the ADC offset in 12-bit single-ended mode:

1. Set ADCOFFTRIM to +112 steps (0x70). This adds an artificial offset to account for negative offset that can 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). Use the maximum value of ACQPS to make sure longer settling time to account for parasitic impedance of internal VREFLO connections.
3. Divide the accumulated result by the multiple of 16 (for example, for 512 conversions, divide by 32).
4. Set ADCOFFTRIM to 112 – result from step 3.

## 11.12 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 make sure that this duration exceeds both 1 ADCCLK period and the minimum S+H duration specified in the data sheet. The conversion time is approximately 10.5 ADCCLK cycles. The exact conversion time is always a whole number of SYSCLK cycles. See the timing diagrams and tables in [Section 11.12.1](#) for exact timings.

### 11.12.1 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).

[Table 11-9](#) describes the parameters in the following timing diagrams. [Table 11-10](#) lists the ADC timings..

**Table 11-9. ADC Timing Parameter Descriptions**

Parameter	Description
$t_{SH}$	<p>The duration of the S+H window.</p> <p>At the end of this window, the value on the S+H capacitor becomes the voltage to be converted into a digital value. The duration is given by <math>(ACQPS + 1)</math> SYSCLK cycles. ACQPS can be configured individually for each SOC, so <math>t_{SH}</math> is not necessarily the same for different SOC's.</p> <p><b>Note:</b> The value on the S+H capacitor is captured approximately 5ns before the end of the S+H window regardless of device clock settings.</p>
$t_{LAT}$	<p>The time from the end of the S+H window until the ADC results latch in the ADCRESULTx register.</p> <p>If the ADCRESULTx register is read before this time, the previous conversion results are returned.</p>
$t_{EOC}$	<p>The time from the end of the S+H window until the S+H window for the next ADC conversion can begin. The subsequent sample can start before the conversion results are latched.</p>
$t_{INT}$	<p>The time from the end of the S+H window until an ADCINT flag is set (if configured).</p> <p>If the INTPULSEPOS bit in the ADCCTL1 register is set, <math>t_{INT}</math> coincides with the end of conversion (EOC) signal.</p> <p>If the INTPULSEPOS bit is 0, and the OFFSET field in the ADCINTCYCLE register is not 0, then there is a delay of OFFSET SYSCLK cycles before the ADCINT flag is set. This delay can be used to enter the ISR exactly when the sample is ready.</p> <p>If the INTPULSEPOS bit is 0, <math>t_{INT}</math> coincides with the end of the S+H window. If <math>t_{INT}</math> triggers a read of the ADC result register (by triggering an ISR that reads the result), care must be taken to make sure the read occurs after the results latch (otherwise, the previous results are read).</p>

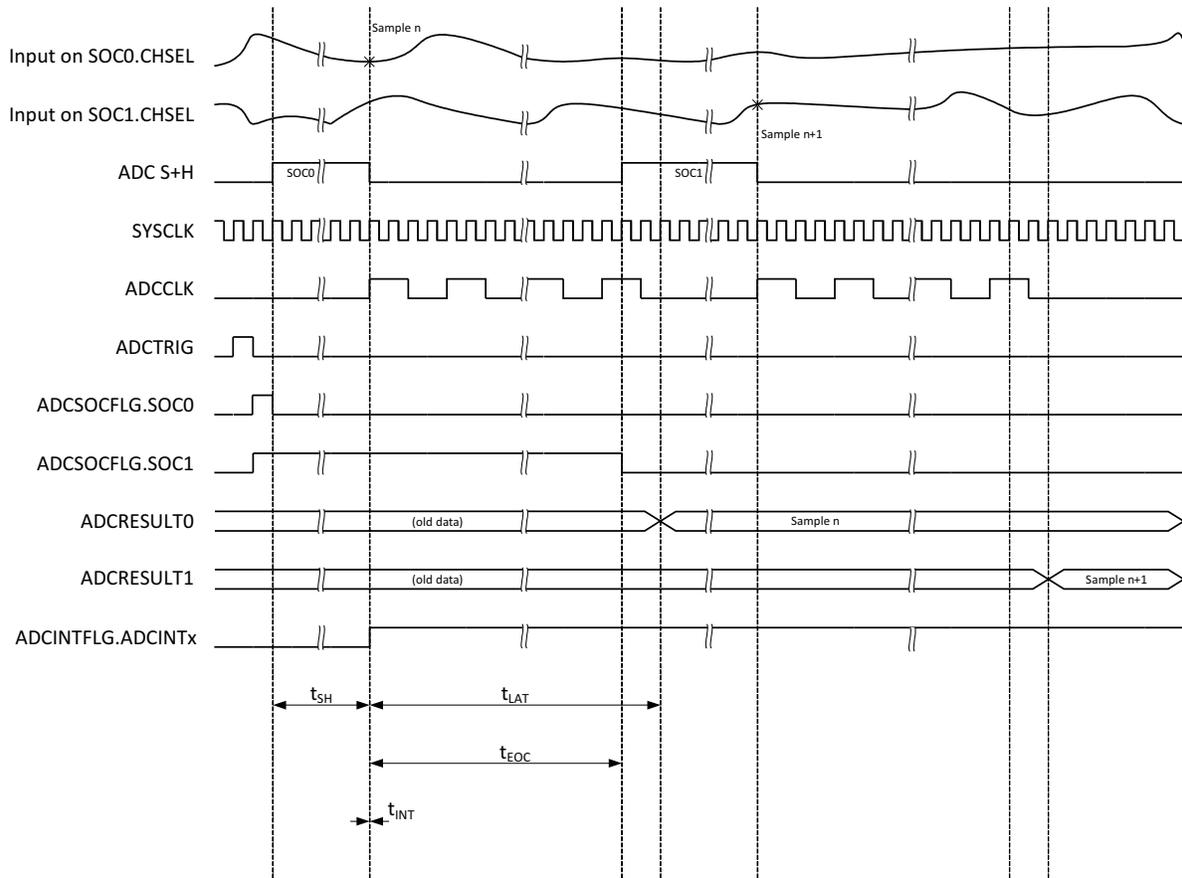
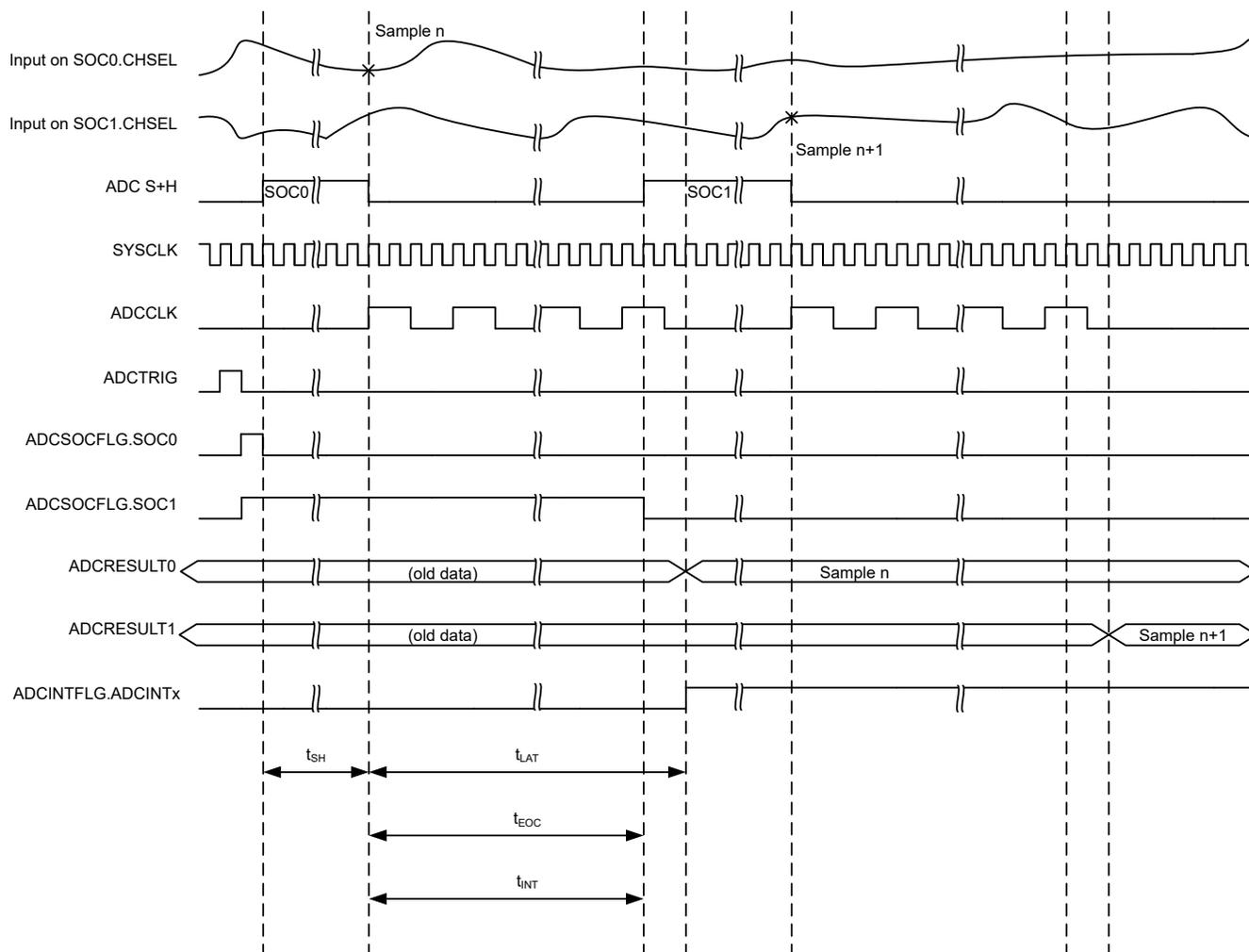


Figure 11-12. ADC Timings for 12-bit Mode in Early Interrupt Mode


**Figure 11-13. ADC Timings for 12-bit Mode in Late Interrupt Mode**
**Table 11-10. ADC Timings in 12-bit Mode**

ADCCLK Prescale		SYSCLK Cycles			
ADCCTL2.PRESCALE	Prescale Ratio	$t_{EOC}$	$t_{LAT}$	$t_{INT}$ (Early) <sup>(1)</sup>	$t_{INT}$ (Late)
0	1	11	13	0	11
2	2	21	23	0	21
4	3	31	34	0	31
6	4	41	44	0	41
8	5	51	55	0	51
10	6	61	65	0	61
12	7	71	76	0	71
14	8	81	86	0	81

(1) By default,  $t_{INT}$  occurs one SYSCLK cycle after the S+H window if INTPULSEPOS is 0. This can be changed by writing to the OFFSET field in the ADCINTCYCLE register.

## 11.13 Additional Information

The following sections contain additional practical information.

### 11.13.1 Ensuring Synchronous Operation

For best performance, all ADCs on the device must be operated synchronously. The device data sheet specifies the performance in both synchronous and asynchronous mode for those parameters which differ between the modes of operation.

To make sure synchronous operation, all ADCs on the device must operate in lockstep. This is accomplished by writing configurations to all ADCs that cause the sampling and conversion phases of all ADCs to be exactly aligned. The easiest way to accomplish this is to write identical values to the SOC configurations for each ADC for trigger select and ACQPS (S+H duration). In addition, synchronous ADCs must also configure identical values for the SOC priority control, burst mode, burst trigger, and burst size.

#### 11.13.1.1 Basic Synchronous Operation

The following example configures two SOC's each on ADCA and ADCC with identical trigger select and ACQPS values. This results in synchronous operation between ADCA and ADCC. For devices with more than two ADCs, the same principles can be used to synchronize all the ADCs.

**Example: Basic Synchronous Operation**

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 converts ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 converts ADCINC0
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 converts ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 converts ADCINC1
AdccRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdccRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB
    
```

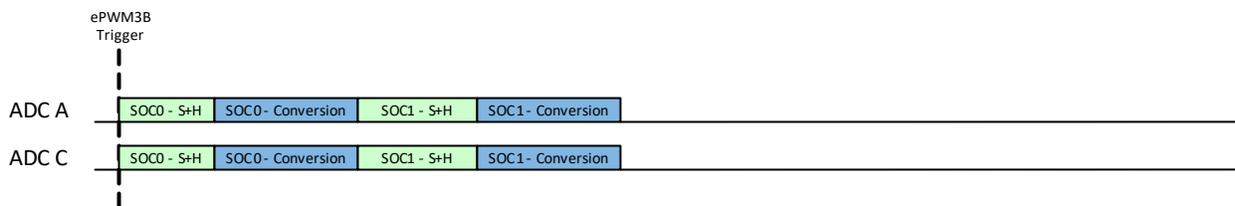


Figure 11-14. Example: Basic Synchronous Operation

Several things can be noted from Figure 11-14. First, while the ACQPS values must be the same for SOC's with the same number, different ACQPS values can be used for SOC's with different numbers. Because of this, synchronous operation does not require a single global S+H time, but instead only channels sampled simultaneously require identical S+H durations. Another important point from this example is that any channel select value can be used for any SOC. Finally, this example assumes round-robin operation. If high-priority SOC's are to be used, the priority must be configured the same on all ADCs.

### 11.13.1.2 Synchronous Operation with Multiple Trigger Sources

As long as each set of SOCs has identical trigger select and ACQPS settings, multiple trigger sources can be used while still achieving synchronous operation.

The following example demonstrates synchronous operation between ADCA and ADCC while using three SOCs and two trigger sources. Figure 11-15 demonstrates that any combination of relative trigger timings still results in synchronous operation.

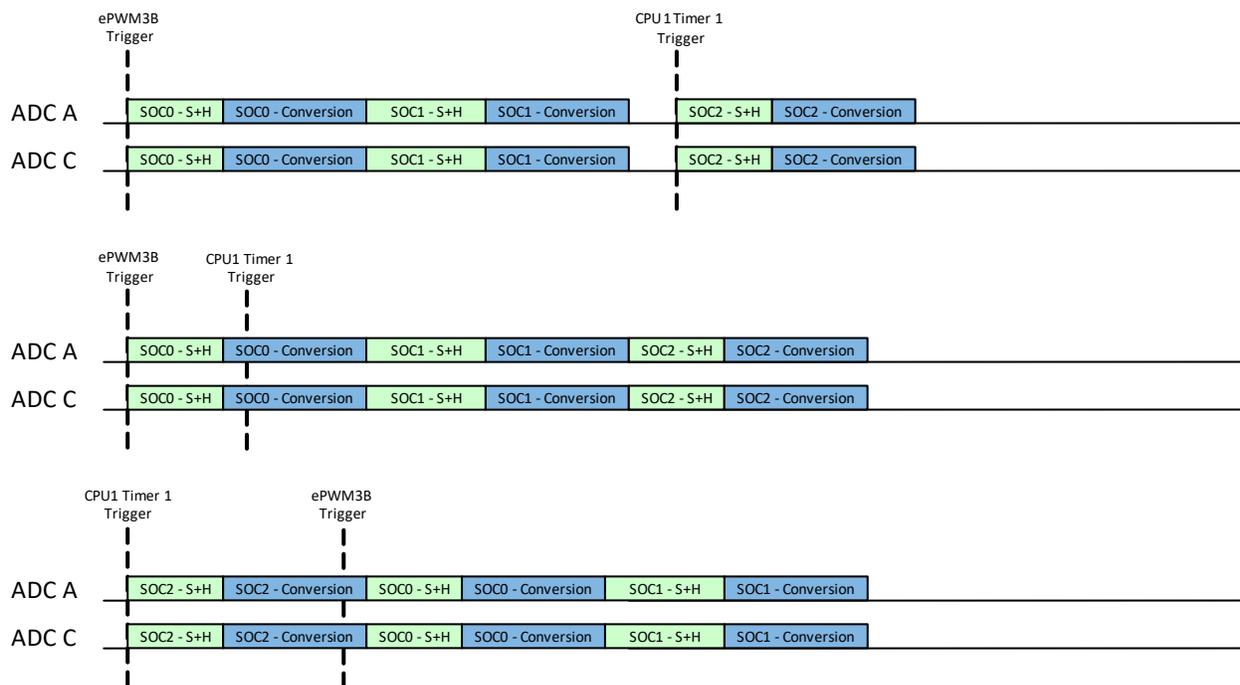
**Example: Synchronous Operation with Multiple Trigger Sources**

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 converts ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 converts ADCINC0
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 converts ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 converts ADCINC1
AdccRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdccRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB

AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0; //SOC2 converts ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 19; //SOC2 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 2; //SOC2 begins conversion on CPU Timer1
AdccRegs.ADCSOC2CTL.bit.CHSEL = 2; //SOC2 converts ADCINC2
AdccRegs.ADCSOC2CTL.bit.ACQPS = 19; //SOC2 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC2CTL.bit.TRIGSEL = 2; //SOC2 begins conversion on CPU Timer1
    
```



**Figure 11-15. Example: Synchronous Operation with Multiple Trigger Sources**

Note that any trigger source that can be selected in the TRIGSEL field can be used except for software triggering. There is no way to issue the software triggers for all ADCs simultaneously, so likely results in asynchronous operation. ADCINT1 or ADCINT2 can also be used as a trigger as long as the ADCINTSOCSEL1 and ADCINTSOCSEL2 registers are configured identically for all ADCs and software triggering is not used to start the chain of conversions.

### 11.13.1.3 Synchronous Operation with Uneven SOC Numbers

If only one trigger source is used, one ADC can use more SOC's than the other ADCs while still operating synchronously.

**Example: Synchronous Operation with Uneven SOC Numbers**

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4; //SOC0 converts ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 0; //SOC0 converts ADCINCO
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19; //SOC0 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 begins conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4; //SOC1 converts ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC1CTL.bit.CHSEL = 1; //SOC1 converts ADCINC1
AdccRegs.ADCSOC1CTL.bit.ACQPS = 30; //SOC1 uses sample duration of 31 SYSCLK cycles
AdccRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 begins conversion on ePWM3 SOCB

AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0; //SOC2 converts ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 30; //SOC2 uses sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 10; //SOC2 begins conversion on ePWM3 SOCB
    
```

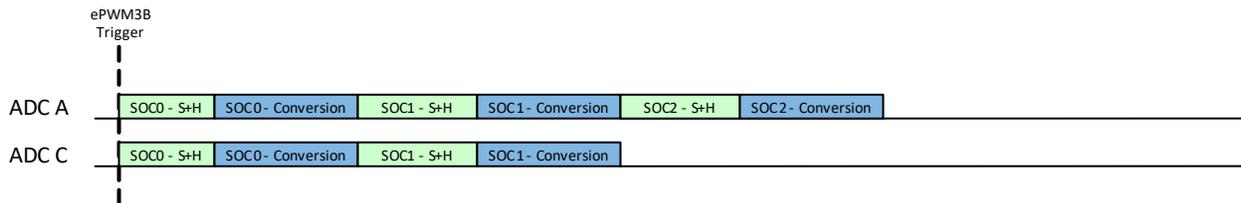


Figure 11-16. Example: Synchronous Operation with Uneven SOC Numbers

Note that if the trigger comes again before all SOC's have completed the conversions, ADCC begins converting immediately on SOC0 while ADCA does not start converting SOC0 again until SOC2 is complete. This results in asynchronous operation, so care must be taken to not overflow the trigger.

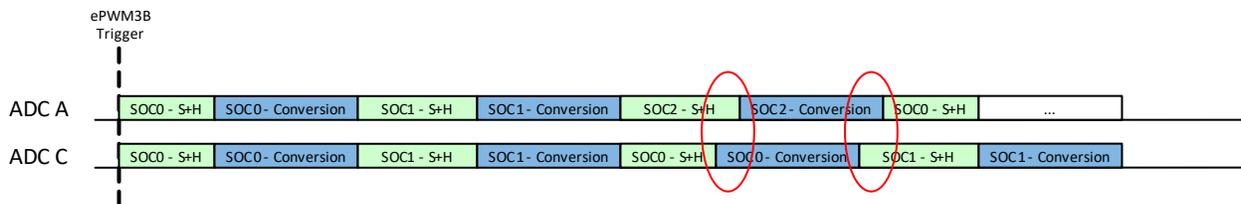


Figure 11-17. Example: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow

### 11.13.1.4 Non-overlapping Conversions

If conversion timings can be made sure to not overlap by the user, then it is not necessary to configure all SOC0s identically on all ADCs to achieve performance equivalent to synchronous operation. For example, if the two ADC triggers in a system come from two ePWM sources that are always 180-degrees out-of-phase, then SOC0 can be used for both ADCA and ADCC with different trigger sources and different ACQPS values.

**Example: Operation with Non-overlapping Conversions**

```
//ePWM3 SOCA and SOCB are 180 degrees out of phase
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;      //SOC0 converts ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;    //SOC0 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10;   //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 0;     //SOC0 converts ADCINC0
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19;    //SOC0 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 9;   //SOC0 begins conversion on ePWM3 SOCA
```

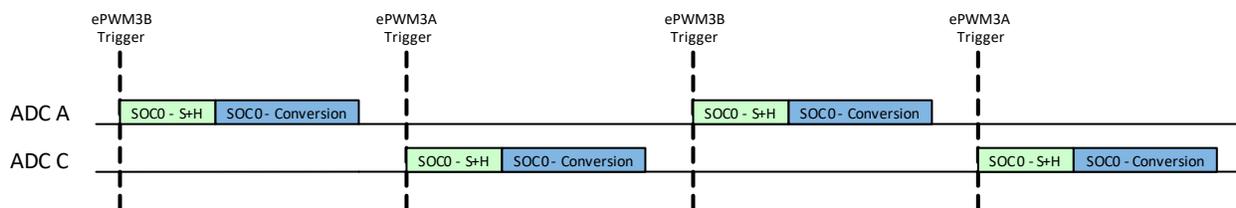


Figure 11-18. Example: Synchronous Equivalent Operation with Non-Overlapping Conversions

### 11.13.2 Choosing an Acquisition Window Duration

For correct operation, the input signal to the ADC must be allowed adequate time to charge the sample and hold capacitor, Ch. Typically, the S+H duration is chosen such that the sampling capacitor is charged to within ½ LSB or ¼ LSB of the final value, depending on the tolerable settling error.

The best methodology to determine the required settling time is to simulate the ADC and ADC driving circuits to make sure adequate settling performance. See [ADC Input Circuit Evaluation for C2000 MCUs](#) and [Charge-Sharing Driving Circuits for C2000 ADCs](#) for additional guidance on ADC signal conditioning circuit design and evaluation.

An approximation of the required settling time can also be determined using an RC settling model. The time constant for the model is given by the equation:

$$\tau = (R_S + R_{on}) \times C_h + R_S \times (C_S + C_p) \quad (5)$$

And the number of time constants needed is given by the equation:

$$k = \ln\left(\frac{2^n}{\text{settling error}}\right) - \ln\left(\frac{C_S + C_P}{CH}\right) \quad (6)$$

So the total S+H time must be set to at least:

$$t = k \cdot \tau \quad (7)$$

Where the following parameters are provided by the ADC input model in the device data sheet:

- $n$  = ADC resolution (in bits)
- $R_{ON}$  = ADC sampling switch resistance (provided in  $\Omega$ )
- $C_H$  = ADC sampling capacitor (provided in pF)
- $C_p$  = ADC channel parasitic input capacitance (provided in pF)

And the following parameters are dependent on the application design:

- settling error = tolerable settling error (in LSBs)
- $R_s$  = ADC driving circuit source impedance (typically in  $\Omega$  or  $k\Omega$ )
- $C_s$  = capacitance on ADC input pin (typically in pF or nF)

For example, assuming the following parameters:

- $n$  = 12-bits
- $R_{ON}$  =  $500\Omega$
- $C_H$  =  $12.5\text{pF}$
- $C_p$  =  $12.7\text{pF}$
- settling error =  $\frac{1}{4}$  LSB
- $R_s$  =  $180\Omega$
- $C_s$  =  $150\text{pF}$

The time constant is calculated as:

$$\tau = (180\Omega + 500\Omega) \times 12.5\text{pF} + 180\Omega \times (150\text{pF} + 12.7\text{pF}) = 37.8\text{ns} \quad (8)$$

And the number of required time constants is:

$$k = \ln\left(\frac{2^{12}}{0.25}\right) - \ln\left(\frac{150\text{pF} + 12.7\text{pF}}{12.5\text{pF}}\right) = 9.70 - 2.57 = 7.13 \quad (9)$$

So the S+H time must be set to at least:  $37.8\text{ns} \times 7.13 = 270\text{ns}$

If  $\text{SYSCLK} = 120\text{MHz}$ , then each  $\text{SYSCLK}$  cycle is  $8.33\text{ns}$ . S+H duration is  $270\text{ns}/8.33\text{ns} = 32.4$   $\text{SYSCLK}$  cycles, so ACQPS for this input is set to at least  $\text{CEILING}(32.4) - 1 = 31$ .

While this gives a rough estimate of the required acquisition window, a better method is 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 data sheet specifies a minimum ADC S+H window duration. Do not use an ACQPS value that gives a duration less than this specification.

---

### 11.13.3 Achieving Simultaneous Sampling

While each ADC does not have dual S+H circuits, achieving simultaneous sampling is accomplished by setting the SOC triggers on two or more ADC modules to use the same trigger source. The following example demonstrates simultaneous sampling on 2 ADCs based on an ePWM3 event. ADCINA3 and ADCINC5 are sampled. An acquisition window of 20 SYSCLK cycles is used, but different durations are possible.

```

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 3;           //SOC0 converts ADCINA3
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;         //SOC0 uses sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10;      //SOC0 begins conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 5;         //SOC0 converts ADCINC5
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19;         //SOC0 uses sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10;      //SOC0 begins conversion on ePWM3 SOCB
    
```

When the ePWM3 trigger is received, all ADCs begin converting in parallel immediately. All results are stored 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 do not happen at exactly the same time.

### 11.13.4 Result Register Mapping

The ADC results and the ADC PPB results are duplicated for each memory bus controller in the system. Bus controllers include all CPUs present on the specific part family and part number. For each bus controller, no access configuration is needed to allow read access to the result registers, and no contention occurs in cases where multiple bus controllers try to read the ADC results simultaneously.

### 11.13.5 Internal Temperature Sensor

The internal temperature sensor measures the junction temperature of the device. The output of the sensor can be sampled with the ADC through an internal connection. This can be enabled on channel ADCIN12 on ADCC and on the CMPSS2\_HP5 input by setting the ENABLE bit in the TSNCTRL register.

To convert the temperature sensor reading into a temperature, pass the temperature sensor reading to the ADC\_getTemperatureC() function in the ADC driverlib.

### 11.13.6 Designing an External Reference Circuit

Figure 11-19 shows the basic organization of the external voltage reference generation circuitry. For best performance, the externally generated reference voltage must be buffered by a precision op-amp with good bandwidth and low output impedance before being driven into the ADC reference pin. A capacitor between the high and low reference pins must be placed on the PCB as close to the pins as practical to help absorb high-frequency currents. A series resistor (typically  $<1\Omega$ ) in series with this capacitor is sometimes necessary to maintain op-amp stability.

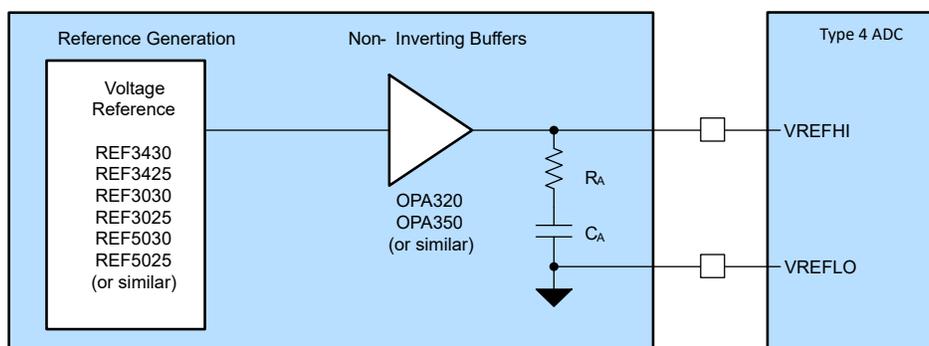


Figure 11-19. ADC Reference System

### 11.13.7 ADC-DAC Loopback Testing

For system diagnostic or functional safety purposes, the user application can perform a loopback test of the ADC module to verify that the ADC is converting correctly. Using the output of the DAC in the first CMPSS module, the device can be configured to supply a series of known voltages to the input of the converter, and the conversion result verified against expected results. Loopback test mode is enabled by setting the bit corresponding to the ADC module under test in the ADCLOOPBACK register in the analog subsystem module to 1. Figure 11-20 shows the connection between the CMPSS DAC output and the ADC.

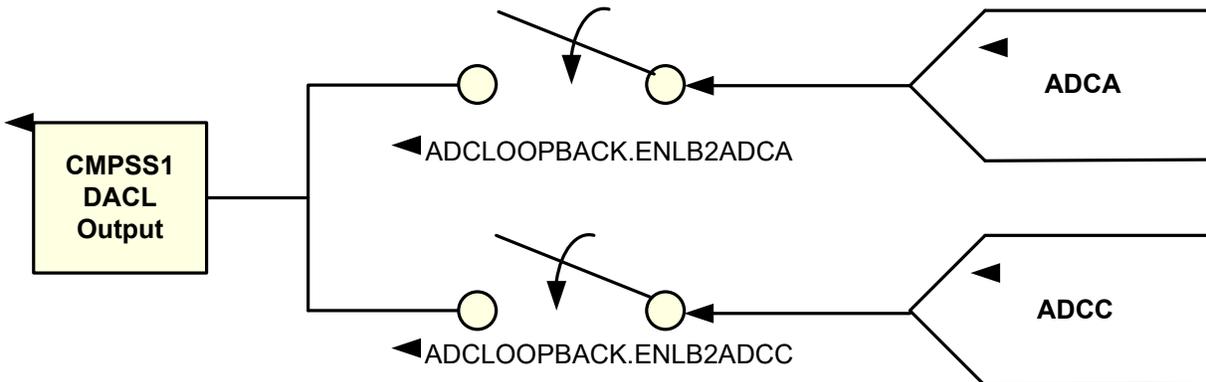


Figure 11-20. CMPSS to ADC Loopback Connection

In ADC loopback test mode, the following special considerations apply:

- The ADC module always samples the CMPSS1 DACL output, regardless of what channel is selected in the ADCSOCxCTL.CHSEL field.
- The minimum sampling window size (ACQPS) when converting the DAC output is 4.27 $\mu$ s (512 SYSCLK cycles at 120MHz SYSCLK) .
- The output resolution of the CMPSS DAC is 6 bits. The lower 6 bits of the input DACVAL are discarded.
- ADC loopback test mode affects CMPSS trip voltages. Avoid enabling ADC loopback mode during regular CMPSS operation.

For more information on the CMPSS module and how to configure the CMPSS DAC, see the *Comparator Subsystem (CMPSS)* chapter.

### 11.13.8 Internal Test Mode

For diagnostic purposes, the ADC can sample various internal node voltages using a special input selection mux called INTERNALTEST. When internal test mode is enabled, the INTERNALTEST mux selection overrides the ADC-A input channel mux: ADC-A samples the INTERNALTEST selection instead of the channel selected by ADCSOCxCTL.CHSEL. Internal test mode can be used to sample the VDDCORE voltage, VREFLO, VDDA, VSSA, and the CMPSS DAC outputs.

To enable internal test mode, write the desired node selection to the TESTSEL field of the INTERNALTESTCTL analog subsystem register. For safety, INTERNALTESTCTL includes a write key field that must be simultaneously written with the value 0xA5A5 for writes to take effect; otherwise, writes to this register are ignored. For details of internal test mux connections to various internal device voltage nodes, refer to the INTERNALTESTCTL register description.

To disable internal test mode, write 0 to the TESTSEL field of the INTERNALTESTCTL register.

When using internal test mode, the following special considerations apply:

- INTERNALTESTCTL.TESTSEL overrides the value of ADCSOCxCTL.CHSEL on ADCA when a non-zero value is configured.
- The minimum sampling window size (ACQPS) when converting INTERNALTEST is 4.27 $\mu$ s (512 SYSCLK cycles at 120MHz SYSCLK).
- The effective resolution of the CMPSS1 DAC outputs to INTERNALTEST is 6 bits.

For more information on the CMPSS module and how to configure the CMPSS DAC, see the *Comparator Subsystem (CMPSS)* chapter.

### 11.13.9 ADC Gain and Offset Calibration

Using the INTERNALTEST mux, gain balancing between multiple ADCs can be performed. By calculating the relative gain error between the ADCs, conversion results can be post-processed in software such that each ADC has the same output for each equivalent input.

To activate gain calibration mode, configure the TESTSEL field of the INTERNALTESTCTL analog subsystem register to select ENZ\_CALIB\_GAIN\_3P3V. In this mode, the voltage sampled by all ADCs when triggered is (VREFHI \* 0.9), overriding the channel selection in ADCSOCxCTL.CHSEL. To turn off gain calibration mode and return to normal operation, configure the TESTSEL field of INTERNALTESTCTL to 0.

## 11.14 Software

### 11.14.1 ADC Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/adc

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](https://dev.ti.com/C2000Ware/Examples).

#### 11.14.1.1 ADC Software Triggering

FILE: adc\_ex1\_soc\_software.c

This example converts some voltages on ADCA and ADCC based on a software trigger.

The ADCC will not convert until ADCA is complete, so the ADCs will not run asynchronously. However, this is much less efficient than allowing the ADCs to convert synchronously in parallel (for example, by using an ePWM trigger).

##### External Connections

- A0, A1, C2, and C3 should be connected to signals to convert

##### Watch Variables

- *myADC0Result0* - Digital representation of the voltage on pin A0
- *myADC0Result1* - Digital representation of the voltage on pin A1
- *myADC1Result0* - Digital representation of the voltage on pin C2
- *myADC1Result1* - Digital representation of the voltage on pin C3

#### 11.14.1.2 ADC ePWM Triggering

FILE: adc\_ex2\_soc\_epwm.c

This example sets up ePWM1 to periodically trigger a conversion on ADCA.

##### External Connections

- A0 should be connected to a signal to convert

##### Watch Variables

- *myADC0Results* - A sequence of analog-to-digital conversion samples from pin A0. The time between samples is determined based on the period of the ePWM timer.

#### 11.14.1.3 ADC Temperature Sensor Conversion

FILE: adc\_ex3\_temp\_sensor.c

This example sets up the ePWM to periodically trigger the ADC. The ADC converts the internal connection to the temperature sensor, which is then interpreted as a temperature by calling the `ADC_getTemperatureC()` function.

##### Watch Variables

- *sensorSample* - The raw reading from the temperature sensor
- *sensorTemp* - The interpretation of the sensor sample as a temperature in degrees Celsius.

#### 11.14.1.4 ADC Synchronous SOC Software Force (*adc\_soc\_software\_sync*)

FILE: adc\_ex4\_soc\_software\_sync.c

This example converts some voltages on ADCA and ADCC using input 5 of the input X-BAR as a software force. Input 5 is triggered by toggling GPIO0, but any spare GPIO could be used. This method will ensure that both ADCs start converting at exactly the same time.

##### External Connections

- A2, A3, C2, C3 pins should be connected to signals to convert

##### Watch Variables

- *myADC0Result0* : a digital representation of the voltage on pin A2
- *myADC0Result1* : a digital representation of the voltage on pin A3
- *myADC1Result0*: a digital representation of the voltage on pin C2
- *myADC1Result1*: a digital representation of the voltage on pin C3

#### 11.14.1.5 ADC Continuous Triggering (*adc\_soc\_continuous*)

FILE: *adc\_ex5\_soc\_continuous.c*

This example sets up the ADC to convert continuously, achieving maximum sampling rate.

##### External Connections

- A0 pin should be connected to signal to convert

##### Watch Variables

- *adcAResults* - A sequence of analog-to-digital conversion samples from pin A0. The time between samples is the minimum possible based on the ADC speed.

#### 11.14.1.6 ADC PPB Offset (*adc\_ppb\_offset*)

FILE: *adc\_ex7\_ppb\_offset.c*

This example software triggers the ADC. Some SOCs have automatic offset adjustment applied by the post-processing block. After the program runs, the memory will contain ADC & post-processing block (PPB) results.

##### External Connections

- A2, C2 pins should be connected to signals to convert

##### Watch Variables

- *myADC0Result* : a digital representation of the voltage on pin A2
- *myADC0PPBResult* : a digital representation of the voltage on pin A2, minus 100 LSBs of automatically added offset
- *myADC1Result*: a digital representation of the voltage on pin C2
- *myADC1PPBResult*: a digital representation of the voltage on pin C2 plus 100 LSBs of automatically added offset

#### 11.14.1.7 ADC PPB Limits (*adc\_ppb\_limits*)

FILE: *adc\_ex8\_ppb\_limits.c*

This example sets up the ePWM to periodically trigger the ADC. If the results are outside of the defined range, the post-processing block will generate an interrupt.

The default limits are 1000LSBs and 3000LSBs. With VREFHI set to 3.3V, the PPB will generate an interrupt if the input voltage goes above about 2.4V or below about 0.8V.

##### External Connections

- A0 should be connected to a signal to convert

##### Watch Variables

- None

#### 11.14.1.8 ADC PPB Delay Capture (*adc\_ppb\_delay*)

FILE: *adc\_ex9\_ppb\_delay.c*

This example demonstrates delay capture using the post-processing block.

Two asynchronous ADC triggers are setup:

- ePWM1, with period 2048, triggering SOC0 to convert on pin A0
  - ePWM2, with period 9999, triggering SOC1 to convert on pin A2
- Each conversion generates an ISR at the end of the conversion. In the ISR for SOC0, a conversion counter is incremented and the PPB is checked to determine if the sample was delayed.

After the program runs, the memory will contain:

*conversion* : the sequence of conversions using SOC0 that were delayed

- *delay* : the corresponding delay of each of the delayed conversions

#### **11.14.1.9 ADC ePWM Triggering Multiple SOC**

FILE: adc\_ex10\_multiple\_soc\_epwm.c

This example sets up ePWM1 to periodically trigger a set of conversions on ADCA and ADCC. This example demonstrates multiple ADCs working together to process a batch of conversions using the available parallelism across multiple ADCs.

ADCA Interrupt ISRs are used to read results of both ADCA and ADCC.

##### *External Connections*

- A0, A1, A2 and C2, C3, C4 pins should be connected to signals to be

##### *Watch Variables*

- *adcAResult0* - Digital representation of the voltage on pin A0
- *adcAResult1* - Digital representation of the voltage on pin A1
- *adcAResult2* - Digital representation of the voltage on pin A2
- *adcCResult0* - Digital representation of the voltage on pin C2
- *adcCResult1* - Digital representation of the voltage on pin C3
- *adcCResult2* - Digital representation of the voltage on pin C4

#### **11.14.1.10 ADC Burst Mode**

FILE: adc\_ex11\_burst\_mode\_epwm.c

This example sets up ePWM1 to periodically trigger ADCA using burst mode. This allows for different channels to be sampled with each burst.

Each burst triggers 3 conversions. A0 and A1 are part of every burst while the third conversion rotates between A2, A3, and A4. This allows high importance signals to be sampled at high speed while lower priority signals can be sampled at a lower rate.

ADCA Interrupt ISRs are used to read results for ADCA.

##### *External Connections*

- A0, A1, A2, A3, A4

##### *Watch Variables*

- *adcAResult0* - Digital representation of the voltage on pin A0
- *adcAResult1* - Digital representation of the voltage on pin A1
- *adcAResult2* - Digital representation of the voltage on pin A2
- *adcAResult3* - Digital representation of the voltage on pin A3
- *adcAResult4* - Digital representation of the voltage on pin A4

#### **11.14.1.11 ADC Burst Mode Oversampling**

FILE: adc\_ex12\_burst\_mode\_oversampling.c

This example is an ADC oversampling example implemented with software. The ADC SOCs are configured in burst mode, triggered by the ePWM SOC A event trigger.

##### *External Connection*

- A2

##### *Watch Variables*

- *Iv\_results* - Array of digital values measured on pin A2 (oversampling is configured by `Oversampling_Amount`)

### 11.14.1.12 ADC SOC Oversampling

FILE: adc\_ex13\_soc\_oversampling.c

This example sets up ePWM1 to periodically trigger a set of conversions on ADCA including multiple SOCs that all convert A2 to achieve oversampling on A2.

ADCA Interrupt ISRs are used to read results of ADCA.

#### External Connections

- A0, A1, A2 should be connected to signals to be converted.

#### Watch Variables

- *adcAResult0* - Digital representation of the voltage on pin A0
- *adcAResult1* - Digital representation of the voltage on pin A1
- *adcAResult2* - Digital representation of the voltage on pin A2

### 11.14.1.13 ADC PPB PWM trip (adc\_ppb\_pwm\_trip)

FILE: adc\_ex14\_ppb\_pwm\_trip.c

This example demonstrates EPWM tripping through ADC limit detection PPB block. ADCAINT1 is configured to periodically trigger the ADCA channel 2 post initial software forced trigger. The limit detection post-processing block(PPB) is configured and if the ADC results are outside of the defined range, the post-processing block will generate an ADCxEVTy event. This event is configured as EPWM trip source through configuring EPWM XBAR and corresponding EPWM's trip zone and digital compare sub-modules. The example showcases

- one-shot
- cycle-by-cycle
- and direct tripping of PWMs through ADCAEVT1 source via Digital compare submodule.

The default limits are 0LSBs and 3600LSBs. With VREFHI set to 3.3V, the PPB will generate a trip event if the input voltage goes above about 2.9V.

#### External Connections

- A2 should be connected to a signal to convert
- Observe the following signals on an oscilloscope
  - ePWM1(GPIO0 - GPIO1)
  - ePWM2(GPIO2 - GPIO3)
  - ePWM3(GPIO4 - GPIO5)

#### Watch Variables

- *adcA2Results* - digital representation of the voltage on pin A2

### 11.14.1.14 ADC Open Shorts Detection (adc\_open\_shorts\_detection)

FILE: adc\_ex15\_open\_shorts\_detection.c

This example demonstrates the ADC open/shorts detection(ADCOSDETECT) circuit configuration for detecting pin faults in the system. The example enables the open/shorts detection circuit along with mandatory ADC configurations and diagnoses ADCA A0 input pin state before starting normal ADC conversions.

To enable the ADC OSDetect circuit:

1. Configure the ADC for conversion (E.g. channel, SOC, ACQPS, prescalar, trigger etc). The OSDetect functionality is available in 12-bit only.
2. Set up the ADCOSDETECT register for the desired voltage divider connection. Refer device TRM for details on available OSDetect configurations.
3. Initiate a conversion and inspect the conversion result. Note: The results must be interpreted based on what is driving on the input side and what are the values of Rs and Cp. If the Vs signal can be disconnected from the input pin, the circuit can be used to detect open and shorted input pins. In the example, ADCA A0 channel is configured and following algorithm is used to check the A0 pin status: Step 1: Configure full

scale OSDETECT mode & capture ADC results(resultHi) Step 2: Configure zero scale OSDETECT mode & capture ADC results(resultLo) Step 3: Disable OSDETECT mode and capture ADC results(resultNormal) Step 4: Determine the state of the ADC pin a. If the pin is open, resultLo would be equal to Vreflo and resultHi would be equal to Vrefhi b. If the pin is shorted to Vrefhi, resultLo should be approximately equal to Vrefhi and resultHi should be equal to Vrefhi c. If the pin is shorted to Vreflo, resultLo should be equal to Vreflo and resultHi should be approximately equal to Vreflo d. If the pin is connected to a valid signal, resultLo should be greater than osdLoLimit but less than resultNormal while resultHi

should be less than osdHiLimit but greater than resultNormal Input | Full-Scale output | Zero-scale Output | Pin Status

Unknown| VREFHI | VREFLO | Open VREFHI | VREFHI | approx. VREFHI | Shorted to VREFHI VREFLO | approx. VREFLO | VREFLO | Shorted to VREFLO

$V_n$  |  $V_n < \text{resultHi} < \text{VREFHI}$  |  $\text{VREFLO} < \text{resultLo} < V_n$  | Good

Step 5: osDetectStatusVal of value greater than 4 would mean that there is no pin fault. a. If osDetectStatusVal == 1, means pin A0 is OPEN b. If osDetectStatusVal == 2, means pin A0 is shorted to VREFLO c. If osDetectStatusVal == 4, means pin A0 is shorted to VREFHI d. If osDetectStatusVal == 8, means pin A0 is in GOOD/VALID state e. Any value of osDetectStatusVal > 4, means pin A0 is in VALID state

Following points should be noted while configuring the ADC in OSDETECT mode.

1. The divider resistance tolerances can vary widely, hence this feature should not be used to check for conversion accuracy.
2. Consult the device data manual for implementation and availability of analog input channels.
3. Due to high drive impedance, a S+H duration much longer than the ADC minimum will be needed.

#### External Connections

- A0 pin should be connected to signals to convert

#### Watch Variables

- *osDetectStatusVal* : OS detection status of voltage on pin A0
- *adcAResult0* : a digital representation of the voltage on pin A0

## 11.15 ADC Registers

This section describes the Analog-to-Digital Converter Registers.

### 11.15.1 ADC Base Address Table

**Table 11-11. ADC Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
AdcaResultRegs	ADC_RESULT_REGS	ADCARERESULT_BASE	0x0000_0B00	-
AdccResultRegs	ADC_RESULT_REGS	ADCCRESULT_BASE	0x0000_0B40	-
AdcaRegs	ADC_REGS	ADCA_BASE	0x0000_7400	YES
AdccRegs	ADC_REGS	ADCC_BASE	0x0000_7500	YES

### 11.15.2 ADC\_RESULT\_REGS Registers

Table 11-12 lists the memory-mapped registers for the ADC\_RESULT\_REGS registers. All register offset addresses not listed in Table 11-12 should be considered as reserved locations and the register contents should not be modified.

**Table 11-12. ADC\_RESULT\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCRESULT0	ADC Result 0 Register		<a href="#">Go</a>
1h	ADCRESULT1	ADC Result 1 Register		<a href="#">Go</a>
2h	ADCRESULT2	ADC Result 2 Register		<a href="#">Go</a>
3h	ADCRESULT3	ADC Result 3 Register		<a href="#">Go</a>
4h	ADCRESULT4	ADC Result 4 Register		<a href="#">Go</a>
5h	ADCRESULT5	ADC Result 5 Register		<a href="#">Go</a>
6h	ADCRESULT6	ADC Result 6 Register		<a href="#">Go</a>
7h	ADCRESULT7	ADC Result 7 Register		<a href="#">Go</a>
8h	ADCRESULT8	ADC Result 8 Register		<a href="#">Go</a>
9h	ADCRESULT9	ADC Result 9 Register		<a href="#">Go</a>
Ah	ADCRESULT10	ADC Result 10 Register		<a href="#">Go</a>
Bh	ADCRESULT11	ADC Result 11 Register		<a href="#">Go</a>
Ch	ADCRESULT12	ADC Result 12 Register		<a href="#">Go</a>
Dh	ADCRESULT13	ADC Result 13 Register		<a href="#">Go</a>
Eh	ADCRESULT14	ADC Result 14 Register		<a href="#">Go</a>
Fh	ADCRESULT15	ADC Result 15 Register		<a href="#">Go</a>
10h	ADCPPB1RESULT	ADC Post Processing Block 1 Result Register		<a href="#">Go</a>
12h	ADCPPB2RESULT	ADC Post Processing Block 2 Result Register		<a href="#">Go</a>
14h	ADCPPB3RESULT	ADC Post Processing Block 3 Result Register		<a href="#">Go</a>
16h	ADCPPB4RESULT	ADC Post Processing Block 4 Result Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 11-13 shows the codes that are used for access types in this section.

**Table 11-13. ADC\_RESULT\_REGS Access Type Codes**

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

### 11.15.2.1 ADCRESULT0 Register (Offset = 0h) [Reset = 0000h]

ADCRESULT0 is shown in [Figure 11-21](#) and described in [Table 11-14](#).

Return to the [Summary Table](#).

ADC Result 0 Register

**Figure 11-21. ADCRESULT0 Register**

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

**Table 11-14. ADCRESULT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 0 16-bit ADC result. After the ADC completes a conversion of SOC0, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.2 ADCRESULT1 Register (Offset = 1h) [Reset = 0000h]

ADCRESULT1 is shown in [Figure 11-22](#) and described in [Table 11-15](#).

Return to the [Summary Table](#).

ADC Result 1 Register

**Figure 11-22. ADCRESULT1 Register**

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

**Table 11-15. ADCRESULT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 1 16-bit ADC result. After the ADC completes a conversion of SOC1, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.3 ADCRESULT2 Register (Offset = 2h) [Reset = 0000h]

ADCRESULT2 is shown in [Figure 11-23](#) and described in [Table 11-16](#).

Return to the [Summary Table](#).

ADC Result 2 Register

**Figure 11-23. ADCRESULT2 Register**

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

**Table 11-16. ADCRESULT2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 2 16-bit ADC result. After the ADC completes a conversion of SOC2, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.4 ADCRESULT3 Register (Offset = 3h) [Reset = 0000h]

ADCRESULT3 is shown in [Figure 11-24](#) and described in [Table 11-17](#).

Return to the [Summary Table](#).

ADC Result 3 Register

**Figure 11-24. ADCRESULT3 Register**

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

**Table 11-17. ADCRESULT3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 3 16-bit ADC result. After the ADC completes a conversion of SOC3, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.5 ADCRESULT4 Register (Offset = 4h) [Reset = 0000h]

ADCRESULT4 is shown in [Figure 11-25](#) and described in [Table 11-18](#).

Return to the [Summary Table](#).

ADC Result 4 Register

**Figure 11-25. ADCRESULT4 Register**

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

**Table 11-18. ADCRESULT4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 4 16-bit ADC result. After the ADC completes a conversion of SOC4, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.6 ADCRESULT5 Register (Offset = 5h) [Reset = 0000h]

ADCRESULT5 is shown in [Figure 11-26](#) and described in [Table 11-19](#).

Return to the [Summary Table](#).

ADC Result 5 Register

**Figure 11-26. ADCRESULT5 Register**

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

**Table 11-19. ADCRESULT5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 5 16-bit ADC result. After the ADC completes a conversion of SOC5, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.7 ADCRESULT6 Register (Offset = 6h) [Reset = 0000h]

ADCRESULT6 is shown in [Figure 11-27](#) and described in [Table 11-20](#).

Return to the [Summary Table](#).

ADC Result 6 Register

**Figure 11-27. ADCRESULT6 Register**

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

**Table 11-20. ADCRESULT6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 6 16-bit ADC result. After the ADC completes a conversion of SOC6, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.8 ADCRESULT7 Register (Offset = 7h) [Reset = 0000h]

ADCRESULT7 is shown in [Figure 11-28](#) and described in [Table 11-21](#).

Return to the [Summary Table](#).

ADC Result 7 Register

**Figure 11-28. ADCRESULT7 Register**

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

**Table 11-21. ADCRESULT7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 7 16-bit ADC result. After the ADC completes a conversion of SOC7, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.9 ADCRESULT8 Register (Offset = 8h) [Reset = 0000h]

ADCRESULT8 is shown in [Figure 11-29](#) and described in [Table 11-22](#).

Return to the [Summary Table](#).

ADC Result 8 Register

**Figure 11-29. ADCRESULT8 Register**

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

**Table 11-22. ADCRESULT8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 8 16-bit ADC result. After the ADC completes a conversion of SOC8, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.10 ADCRESULT9 Register (Offset = 9h) [Reset = 0000h]

ADCRESULT9 is shown in [Figure 11-30](#) and described in [Table 11-23](#).

Return to the [Summary Table](#).

ADC Result 9 Register

**Figure 11-30. ADCRESULT9 Register**

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

**Table 11-23. ADCRESULT9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 9 16-bit ADC result. After the ADC completes a conversion of SOC9, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.11 ADCRESULT10 Register (Offset = Ah) [Reset = 0000h]

ADCRESULT10 is shown in [Figure 11-31](#) and described in [Table 11-24](#).

Return to the [Summary Table](#).

ADC Result 10 Register

**Figure 11-31. ADCRESULT10 Register**

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

**Table 11-24. ADCRESULT10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 10 16-bit ADC result. After the ADC completes a conversion of SOC10, the digital result is placed in this bit field. Reset type: SYSRSn

**11.15.2.12 ADCRESULT11 Register (Offset = Bh) [Reset = 0000h]**

ADCRESULT11 is shown in [Figure 11-32](#) and described in [Table 11-25](#).

Return to the [Summary Table](#).

ADC Result 11 Register

**Figure 11-32. ADCRESULT11 Register**

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

**Table 11-25. ADCRESULT11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 11 16-bit ADC result. After the ADC completes a conversion of SOC11, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.13 ADCRESULT12 Register (Offset = Ch) [Reset = 0000h]

ADCRESULT12 is shown in [Figure 11-33](#) and described in [Table 11-26](#).

Return to the [Summary Table](#).

ADC Result 12 Register

**Figure 11-33. ADCRESULT12 Register**

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

**Table 11-26. ADCRESULT12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 12 16-bit ADC result. After the ADC completes a conversion of SOC12, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.14 ADCRESULT13 Register (Offset = Dh) [Reset = 0000h]

ADCRESULT13 is shown in [Figure 11-34](#) and described in [Table 11-27](#).

Return to the [Summary Table](#).

ADC Result 13 Register

**Figure 11-34. ADCRESULT13 Register**

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

**Table 11-27. ADCRESULT13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 13 16-bit ADC result. After the ADC completes a conversion of SOC13, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.15 ADCRESULT14 Register (Offset = Eh) [Reset = 0000h]

ADCRESULT14 is shown in [Figure 11-35](#) and described in [Table 11-28](#).

Return to the [Summary Table](#).

ADC Result 14 Register

**Figure 11-35. ADCRESULT14 Register**

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

**Table 11-28. ADCRESULT14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 14 16-bit ADC result. After the ADC completes a conversion of SOC14, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.16 ADCRESULT15 Register (Offset = Fh) [Reset = 0000h]

ADCRESULT15 is shown in [Figure 11-36](#) and described in [Table 11-29](#).

Return to the [Summary Table](#).

ADC Result 15 Register

**Figure 11-36. ADCRESULT15 Register**

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

**Table 11-29. ADCRESULT15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RESULT	R	0h	ADC Result 15 16-bit ADC result. After the ADC completes a conversion of SOC15, the digital result is placed in this bit field. Reset type: SYSRSn

### 11.15.2.17 ADCPPB1RESULT Register (Offset = 10h) [Reset = 0000000h]

ADCPPB1RESULT is shown in [Figure 11-37](#) and described in [Table 11-30](#).

Return to the [Summary Table](#).

ADC Post Processing Block 1 Result Register

**Figure 11-37. ADCPPB1RESULT Register**

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

**Table 11-30. ADCPPB1RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	Sign Extended Bits. These bits reflect the same value as bit 16. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 1 The result of the offset/reference subtraction post conversion processing is stored in this register. This result is available 1 SYSCLK cycle after the associated ADCRESULT is available. If ADCINTFLG is polled to determine when to read the PPBRESULT, it may be necessary to add a NOP instruction to ensure that the updated post conversion processing result has posted to the register. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn

### 11.15.2.18 ADCPPB2RESULT Register (Offset = 12h) [Reset = 0000000h]

ADCPPB2RESULT is shown in [Figure 11-38](#) and described in [Table 11-31](#).

Return to the [Summary Table](#).

ADC Post Processing Block 2 Result Register

**Figure 11-38. ADCPPB2RESULT Register**

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

**Table 11-31. 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 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 2 The result of the offset/reference subtraction post conversion processing is stored in this register. This result is available 1 SYSCLK cycle after the associated ADCRESULT is available. If ADCINTFLG is polled to determine when to read the PPBRESULT, it may be necessary to add a NOP instruction to ensure that the updated post conversion processing result has posted to the register. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn

### 11.15.2.19 ADCPPB3RESULT Register (Offset = 14h) [Reset = 0000000h]

ADCPPB3RESULT is shown in [Figure 11-39](#) and described in [Table 11-32](#).

Return to the [Summary Table](#).

ADC Post Processing Block 3 Result Register

**Figure 11-39. ADCPPB3RESULT Register**

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

**Table 11-32. ADCPPB3RESULT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	SIGN	R	0h	Sign Extended Bits. These bits reflect the same value as bit 16. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the SIGN bits extend down to bit 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 3 The result of the offset/reference subtraction post conversion processing is stored in this register. This result is available 1 SYSCLK cycle after the associated ADCRESULT is available. If ADCINTFLG is polled to determine when to read the PPBRESULT, it may be necessary to add a NOP instruction to ensure that the updated post conversion processing result has posted to the register. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn

### 11.15.2.20 ADCPPB4RESULT Register (Offset = 16h) [Reset = 0000000h]

ADCPPB4RESULT is shown in [Figure 11-40](#) and described in [Table 11-33](#).

Return to the [Summary Table](#).

ADC Post Processing Block 4 Result Register

**Figure 11-40. ADCPPB4RESULT Register**

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

**Table 11-33. 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 12, and all reflect the same value as bit 12. Reset type: SYSRSn
15-0	PPBRESULT	R	0h	ADC Post Processing Block Result 4 The result of the offset/reference subtraction post conversion processing is stored in this register. This result is available 1 SYSCLK cycle after the associated ADCRESULT is available. If ADCINTFLG is polled to determine when to read the PPBRESULT, it may be necessary to add a NOP instruction to ensure that the updated post conversion processing result has posted to the register. NOTE: If the conversion associated with this Post Processing Block is a 12-bit conversion, the PPBRESULT bits are limited to bits 12:0. Reset type: SYSRSn

### 11.15.3 ADC\_REGS Registers

Table 11-34 lists the memory-mapped registers for the ADC\_REGS registers. All register offset addresses not listed in Table 11-34 should be considered as reserved locations and the register contents should not be modified.

**Table 11-34. ADC\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	ADCCTL1	ADC Control 1 Register	EALLOW	<a href="#">Go</a>
1h	ADCCTL2	ADC Control 2 Register	EALLOW	<a href="#">Go</a>
2h	ADCBURSTCTL	ADC Burst Control Register	EALLOW	<a href="#">Go</a>
3h	ADCINTFLG	ADC Interrupt Flag Register		<a href="#">Go</a>
4h	ADCINTFLGCLR	ADC Interrupt Flag Clear Register		<a href="#">Go</a>
5h	ADCINTOVF	ADC Interrupt Overflow Register		<a href="#">Go</a>
6h	ADCINTOVFCLR	ADC Interrupt Overflow Clear Register		<a href="#">Go</a>
7h	ADCINTSEL1N2	ADC Interrupt 1 and 2 Selection Register	EALLOW	<a href="#">Go</a>
8h	ADCINTSEL3N4	ADC Interrupt 3 and 4 Selection Register	EALLOW	<a href="#">Go</a>
9h	ADCSOCPRICTL	ADC SOC Priority Control Register	EALLOW	<a href="#">Go</a>
Ah	ADCINTSOCSEL1	ADC Interrupt SOC Selection 1 Register	EALLOW	<a href="#">Go</a>
Bh	ADCINTSOCSEL2	ADC Interrupt SOC Selection 2 Register	EALLOW	<a href="#">Go</a>
Ch	ADCSOCFLG1	ADC SOC Flag 1 Register		<a href="#">Go</a>
Dh	ADCSOCFRC1	ADC SOC Force 1 Register		<a href="#">Go</a>
Eh	ADCSOCOVF1	ADC SOC Overflow 1 Register		<a href="#">Go</a>
Fh	ADCSOCOVFCLR1	ADC SOC Overflow Clear 1 Register		<a href="#">Go</a>
10h	ADCSOC0CTL	ADC SOC0 Control Register	EALLOW	<a href="#">Go</a>
12h	ADCSOC1CTL	ADC SOC1 Control Register	EALLOW	<a href="#">Go</a>
14h	ADCSOC2CTL	ADC SOC2 Control Register	EALLOW	<a href="#">Go</a>
16h	ADCSOC3CTL	ADC SOC3 Control Register	EALLOW	<a href="#">Go</a>
18h	ADCSOC4CTL	ADC SOC4 Control Register	EALLOW	<a href="#">Go</a>
1Ah	ADCSOC5CTL	ADC SOC5 Control Register	EALLOW	<a href="#">Go</a>
1Ch	ADCSOC6CTL	ADC SOC6 Control Register	EALLOW	<a href="#">Go</a>
1Eh	ADCSOC7CTL	ADC SOC7 Control Register	EALLOW	<a href="#">Go</a>
20h	ADCSOC8CTL	ADC SOC8 Control Register	EALLOW	<a href="#">Go</a>
22h	ADCSOC9CTL	ADC SOC9 Control Register	EALLOW	<a href="#">Go</a>
24h	ADCSOC10CTL	ADC SOC10 Control Register	EALLOW	<a href="#">Go</a>
26h	ADCSOC11CTL	ADC SOC11 Control Register	EALLOW	<a href="#">Go</a>
28h	ADCSOC12CTL	ADC SOC12 Control Register	EALLOW	<a href="#">Go</a>
2Ah	ADCSOC13CTL	ADC SOC13 Control Register	EALLOW	<a href="#">Go</a>
2Ch	ADCSOC14CTL	ADC SOC14 Control Register	EALLOW	<a href="#">Go</a>
2Eh	ADCSOC15CTL	ADC SOC15 Control Register	EALLOW	<a href="#">Go</a>
30h	ADCEVTSTAT	ADC Event Status Register		<a href="#">Go</a>
32h	ADCEVTCLR	ADC Event Clear Register		<a href="#">Go</a>
34h	ADCEVTSEL	ADC Event Selection Register	EALLOW	<a href="#">Go</a>
36h	ADCEVTINTSEL	ADC Event Interrupt Selection Register	EALLOW	<a href="#">Go</a>
38h	ADCOSDETECT	ADC Open and Shorts Detect Register	EALLOW	<a href="#">Go</a>
39h	ADCCOUNTER	ADC Counter Register		<a href="#">Go</a>
3Ah	ADCREV	ADC Revision Register		<a href="#">Go</a>
3Bh	ADCOFFTRIM	ADC Offset Trim Register	EALLOW	<a href="#">Go</a>

**Table 11-34. ADC\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
40h	ADCPPB1CONFIG	ADC PPB1 Config Register	EALLOW	<a href="#">Go</a>
41h	ADCPPB1STAMP	ADC PPB1 Sample Delay Time Stamp Register		<a href="#">Go</a>
42h	ADCPPB1OFFCAL	ADC PPB1 Offset Calibration Register	EALLOW	<a href="#">Go</a>
43h	ADCPPB1OFFREF	ADC PPB1 Offset Reference Register		<a href="#">Go</a>
44h	ADCPPB1TRIPHI	ADC PPB1 Trip High Register	EALLOW	<a href="#">Go</a>
46h	ADCPPB1TRIPLO	ADC PPB1 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
48h	ADCPPB2CONFIG	ADC PPB2 Config Register	EALLOW	<a href="#">Go</a>
49h	ADCPPB2STAMP	ADC PPB2 Sample Delay Time Stamp Register		<a href="#">Go</a>
4Ah	ADCPPB2OFFCAL	ADC PPB2 Offset Calibration Register	EALLOW	<a href="#">Go</a>
4Bh	ADCPPB2OFFREF	ADC PPB2 Offset Reference Register		<a href="#">Go</a>
4Ch	ADCPPB2TRIPHI	ADC PPB2 Trip High Register	EALLOW	<a href="#">Go</a>
4Eh	ADCPPB2TRIPLO	ADC PPB2 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
50h	ADCPPB3CONFIG	ADC PPB3 Config Register	EALLOW	<a href="#">Go</a>
51h	ADCPPB3STAMP	ADC PPB3 Sample Delay Time Stamp Register		<a href="#">Go</a>
52h	ADCPPB3OFFCAL	ADC PPB3 Offset Calibration Register	EALLOW	<a href="#">Go</a>
53h	ADCPPB3OFFREF	ADC PPB3 Offset Reference Register		<a href="#">Go</a>
54h	ADCPPB3TRIPHI	ADC PPB3 Trip High Register	EALLOW	<a href="#">Go</a>
56h	ADCPPB3TRIPLO	ADC PPB3 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
58h	ADCPPB4CONFIG	ADC PPB4 Config Register	EALLOW	<a href="#">Go</a>
59h	ADCPPB4STAMP	ADC PPB4 Sample Delay Time Stamp Register		<a href="#">Go</a>
5Ah	ADCPPB4OFFCAL	ADC PPB4 Offset Calibration Register	EALLOW	<a href="#">Go</a>
5Bh	ADCPPB4OFFREF	ADC PPB4 Offset Reference Register		<a href="#">Go</a>
5Ch	ADCPPB4TRIPHI	ADC PPB4 Trip High Register	EALLOW	<a href="#">Go</a>
5Eh	ADCPPB4TRIPLO	ADC PPB4 Trip Low/Trigger Time Stamp Register	EALLOW	<a href="#">Go</a>
6Fh	ADCINTCYCLE	ADC Early Interrupt Generation Cycle	EALLOW	<a href="#">Go</a>
72h	ADCINLTRIM2	ADC Linearity Trim 2 Register	EALLOW	<a href="#">Go</a>
74h	ADCINLTRIM3	ADC Linearity Trim 3 Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 11-35](#) shows the codes that are used for access types in this section.

**Table 11-35. ADC\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		

**Table 11-35. ADC\_REGS Access Type Codes (continued)**

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

### 11.15.3.1 ADCCTL1 Register (Offset = 0h) [Reset = 0000h]

ADCCTL1 is shown in [Figure 11-41](#) and described in [Table 11-36](#).

Return to the [Summary Table](#).

ADC Control 1 Register

**Figure 11-41. ADCCTL1 Register**

15	14	13	12	11	10	9	8
RESERVED		ADCBSY	RESERVED	ADCBSYCHN			
R-0h		R-0h	R-0h	R-0h			
7	6	5	4	3	2	1	0
ADCPWDNZ	RESERVED				INTPULSEPOS	RESERVED	
R/W-0h	R-0h			R/W-0h		R-0h	

**Table 11-36. ADCCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13	ADCBSY	R	0h	ADC Busy. Set when ADC SOC is generated, cleared by hardware four ADC clocks after negative edge of S/H pulse. Used by the ADC state machine to determine if ADC is available to sample. 0 ADC is available to sample next channel 1 ADC is busy and cannot sample another channel Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	ADCBSYCHN	R	0h	ADC Busy Channel. Set when an ADC Start of Conversion (SOC) is generated. When ADCBSY=0: holds the value of the last converted SOC When ADCBSY=1: reflects the SOC currently being processed 0h SOC0 is currently processing or was last SOC converted 1h SOC1 is currently processing or was last SOC converted 2h SOC2 is currently processing or was last SOC converted 3h SOC3 is currently processing or was last SOC converted 4h SOC4 is currently processing or was last SOC converted 5h SOC5 is currently processing or was last SOC converted 6h SOC6 is currently processing or was last SOC converted 7h SOC7 is currently processing or was last SOC converted 8h SOC8 is currently processing or was last SOC converted 9h SOC9 is currently processing or was last SOC converted Ah SOC10 is currently processing or was last SOC converted Bh SOC11 is currently processing or was last SOC converted Ch SOC12 is currently processing or was last SOC converted Dh SOC13 is currently processing or was last SOC converted Eh SOC14 is currently processing or was last SOC converted Fh SOC15 is currently processing or was last SOC converted Reset type: SYSRSn
7	ADCPWDNZ	R/W	0h	ADC Power Down (active low). This bit controls the power up and power down of all the analog circuitry inside the analog core. 0 All analog circuitry inside the core is powered down 1 All analog circuitry inside the core is powered up Reset type: SYSRSn
6-3	RESERVED	R	0h	Reserved
2	INTPULSEPOS	R/W	0h	ADC Interrupt Pulse Position. 0 Interrupt pulse generation occurs when ADC begins conversion (at the end of the acquisition window) plus a number of SYSCLK cycles as specified in the ADCINTCYCLE.OFFSET register. 1 Interrupt pulse generation occurs at the end of the conversion, 1 cycle prior to the ADC result latching into its result register Reset type: SYSRSn

**Table 11-36. ADCCTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	RESERVED	R	0h	Reserved

### 11.15.3.2 ADCCTL2 Register (Offset = 1h) [Reset = 0000h]

ADCCTL2 is shown in [Figure 11-42](#) and described in [Table 11-37](#).

Return to the [Summary Table](#).

ADC Control 2 Register

**Figure 11-42. ADCCTL2 Register**

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

**Table 11-37. ADCCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	RESERVED	R	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5-4	RESERVED	R	0h	Reserved
3-0	PRESCALE	R/W	0h	ADC Clock Prescaler. 0000 ADCCLK = Input Clock / 1.0 0001 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 Reset type: SYSRSn

### 11.15.3.3 ADCBURSTCTL Register (Offset = 2h) [Reset = 0000h]

ADCBURSTCTL is shown in [Figure 11-43](#) and described in [Table 11-38](#).

Return to the [Summary Table](#).

ADC Burst Control Register

**Figure 11-43. ADCBURSTCTL Register**

15	14	13	12	11	10	9	8
BURSTEN	RESERVED			BURSTSIZE			
R/W-0h	R-0h			R/W-0h			
7	6	5	4	3	2	1	0
RESERVED		BURSTTRIGSEL					
R-0h		R/W-0h					

**Table 11-38. ADCBURSTCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	BURSTEN	R/W	0h	SOC Burst Mode Enable. This bit enables the SOC Burst Mode of operation. 0 Burst mode is disabled. 1 Burst mode is enabled. Reset type: SYSRSn
14-12	RESERVED	R	0h	Reserved
11-8	BURSTSIZE	R/W	0h	SOC Burst Size Select. This bit field determines how many SOC'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 Reset type: SYSRSn
7-6	RESERVED	R	0h	Reserved

**Table 11-38. ADCBURSTCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-0	BURSTTRIGSEL	R/W	0h	SOC Burst Trigger Source Select. Configures which trigger will start a burst conversion sequence. Note: SOCFRC1 register can always be used to software trigger SOC's in addition to any hardware trigger configuration. 00h BURSTTRIG0 - Software only 01h BURSTTRIG1 - CPU1 Timer 0, TINT0n 02h BURSTTRIG2 - CPU1 Timer 1, TINT1n 03h BURSTTRIG3 - CPU1 Timer 2, TINT2n 04h BURSTTRIG4 - GPIO, Input X-Bar INPUT5 05h BURSTTRIG5 - ePWM1, ADCSOCA 06h BURSTTRIG6 - ePWM1, ADCSOCB 07h BURSTTRIG7 - ePWM2, ADCSOCA 08h BURSTTRIG8 - ePWM2, ADCSOCB 09h BURSTTRIG9 - ePWM3, ADCSOCA 0Ah BURSTTRIG10 - ePWM3, ADCSOCB 0Bh BURSTTRIG11 - ePWM4, ADCSOCA 0Ch BURSTTRIG12 - ePWM4, ADCSOCB 0Dh BURSTTRIG13 - ePWM5, ADCSOCA 0Eh BURSTTRIG14 - ePWM5, ADCSOCB 0Fh BURSTTRIG15 - ePWM6, ADCSOCA 10h BURSTTRIG16 - ePWM6, ADCSOCB 11h BURSTTRIG17 - ePWM7, ADCSOCA 12h BURSTTRIG18 - ePWM7, ADCSOCB 13h - 3Fh - Reserved Reset type: SYSRSn

### 11.15.3.4 ADCINTFLG Register (Offset = 3h) [Reset = 0000h]

ADCINTFLG is shown in [Figure 11-44](#) and described in [Table 11-39](#).

Return to the [Summary Table](#).

ADC Interrupt Flag Register

**Figure 11-44. ADCINTFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 11-39. 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 continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINTFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register. Reset type: SYSRSn
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 continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINTFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register. Reset type: SYSRSn
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 continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINTFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register. Reset type: SYSRSn

**Table 11-39. ADCINTFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	ADCINT1	R	0h	ADC Interrupt 1 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 continue to interrupt mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINTFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register. Reset type: SYSRSn

### 11.15.3.5 ADCINTFLGCLR Register (Offset = 4h) [Reset = 0000h]

ADCINTFLGCLR is shown in [Figure 11-45](#) and described in [Table 11-40](#).

Return to the [Summary Table](#).

ADC Interrupt Flag Clear Register

**Figure 11-45. 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/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h

**Table 11-40. ADCINTFLGCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R-0/W1C	0h	ADC Interrupt 4 Flag Clear. Reads return 0. 0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority and the overflow bit will not be set Reset type: SYSRSn
2	ADCINT3	R-0/W1C	0h	ADC Interrupt 3 Flag Clear. Reads return 0. 0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority and the overflow bit will not be set Reset type: SYSRSn
1	ADCINT2	R-0/W1C	0h	ADC Interrupt 2 Flag Clear. Reads return 0. 0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority and the overflow bit will not be set Reset type: SYSRSn
0	ADCINT1	R-0/W1C	0h	ADC Interrupt 1 Flag Clear. Reads return 0. 0 No action 1 Clears respective flag bit in the ADCINTFLG register. If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority and the overflow bit will not be set Reset type: SYSRSn

### 11.15.3.6 ADCINTOVF Register (Offset = 5h) [Reset = 0000h]

ADCINTOVF is shown in [Figure 11-46](#) and described in [Table 11-41](#).

Return to the [Summary Table](#).

ADC Interrupt Overflow Register

**Figure 11-46. ADCINTOVF Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ADCINT4	ADCINT3	ADCINT2	ADCINT1
R-0h				R-0h	R-0h	R-0h	R-0h

**Table 11-41. ADCINTOVF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R	0h	ADC Interrupt 4 Overflow Flags Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs. 0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected. The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn
2	ADCINT3	R	0h	ADC Interrupt 3 Overflow Flags Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs. 0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected. The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn
1	ADCINT2	R	0h	ADC Interrupt 2 Overflow Flags Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs. 0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected. The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn
0	ADCINT1	R	0h	ADC Interrupt 1 Overflow Flags Indicates if an overflow occurred when generating ADCINT pulses. If the respective ADCINTFLG bit is set and a selected additional EOC trigger is generated, then an overflow condition occurs. 0 No ADC interrupt overflow event detected. 1 ADC Interrupt overflow event detected. The overflow bit does not care about the continuous mode bit state. An overflow condition is generated irrespective of this mode selection. Reset type: SYSRSn

### 11.15.3.7 ADCINTOVFCLR Register (Offset = 6h) [Reset = 0000h]

ADCINTOVFCLR is shown in [Figure 11-47](#) and described in [Table 11-42](#).

Return to the [Summary Table](#).

ADC Interrupt Overflow Clear Register

**Figure 11-47. 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/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h

**Table 11-42. ADCINTOVFCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	ADCINT4	R-0/W1C	0h	ADC Interrupt 4 Overflow Clear Bits 0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
2	ADCINT3	R-0/W1C	0h	ADC Interrupt 3 Overflow Clear Bits 0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
1	ADCINT2	R-0/W1C	0h	ADC Interrupt 2 Overflow Clear Bits 0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn
0	ADCINT1	R-0/W1C	0h	ADC Interrupt 1 Overflow Clear Bits 0 No action. 1 Clears the respective overflow bit in the ADCINTOVF register. If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCINTOVF register, then hardware has priority and the ADCINTOVF bit will be set. Reset type: SYSRSn

### 11.15.3.8 ADCINTSEL1N2 Register (Offset = 7h) [Reset = 0000h]

ADCINTSEL1N2 is shown in [Figure 11-48](#) and described in [Table 11-43](#).

Return to the [Summary Table](#).

ADC Interrupt 1 and 2 Selection Register

**Figure 11-48. ADCINTSEL1N2 Register**

15	14	13	12	11	10	9	8
RESERVED	INT2CONT	INT2E	RESERVED	INT2SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	INT1CONT	INT1E	RESERVED	INT1SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			

**Table 11-43. ADCINTSEL1N2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT2CONT	R/W	0h	ADCINT2 Continue to Interrupt Mode 0 No further ADCINT2 pulses are generated until ADCINT2 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT2 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
13	INT2E	R/W	0h	ADCINT2 Interrupt Enable 0 ADCINT2 is disabled 1 ADCINT2 is enabled Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	INT2SEL	R/W	0h	ADCINT2 EOC Source Select 0h EOC0 is trigger for ADCINT2 1h EOC1 is trigger for ADCINT2 2h EOC2 is trigger for ADCINT2 3h EOC3 is trigger for ADCINT2 4h EOC4 is trigger for ADCINT2 5h EOC5 is trigger for ADCINT2 6h EOC6 is trigger for ADCINT2 7h EOC7 is trigger for ADCINT2 8h EOC8 is trigger for ADCINT2 9h EOC9 is trigger for ADCINT2 Ah EOC10 is trigger for ADCINT2 Bh EOC11 is trigger for ADCINT2 Ch EOC12 is trigger for ADCINT2 Dh EOC13 is trigger for ADCINT2 Eh EOC14 is trigger for ADCINT2 Fh EOC15 is trigger for ADCINT2 Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	INT1CONT	R/W	0h	ADCINT1 Continue to Interrupt Mode 0 No further ADCINT1 pulses are generated until ADCINT1 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT1 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
5	INT1E	R/W	0h	ADCINT1 Interrupt Enable 0 ADCINT1 is disabled 1 ADCINT1 is enabled Reset type: SYSRSn

**Table 11-43. ADCINTSEL1N2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R	0h	Reserved
3-0	INT1SEL	R/W	0h	ADCINT1 EOC Source Select 0h EOC0 is trigger for ADCINT1 1h EOC1 is trigger for ADCINT1 2h EOC2 is trigger for ADCINT1 3h EOC3 is trigger for ADCINT1 4h EOC4 is trigger for ADCINT1 5h EOC5 is trigger for ADCINT1 6h EOC6 is trigger for ADCINT1 7h EOC7 is trigger for ADCINT1 8h EOC8 is trigger for ADCINT1 9h EOC9 is trigger for ADCINT1 Ah EOC10 is trigger for ADCINT1 Bh EOC11 is trigger for ADCINT1 Ch EOC12 is trigger for ADCINT1 Dh EOC13 is trigger for ADCINT1 Eh EOC14 is trigger for ADCINT1 Fh EOC15 is trigger for ADCINT1 Reset type: SYSRStn

### 11.15.3.9 ADCINTSEL3N4 Register (Offset = 8h) [Reset = 0000h]

ADCINTSEL3N4 is shown in [Figure 11-49](#) and described in [Table 11-44](#).

Return to the [Summary Table](#).

ADC Interrupt 3 and 4 Selection Register

**Figure 11-49. ADCINTSEL3N4 Register**

15	14	13	12	11	10	9	8
RESERVED	INT4CONT	INT4E	RESERVED	INT4SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	INT3CONT	INT3E	RESERVED	INT3SEL			
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h			

**Table 11-44. ADCINTSEL3N4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	INT4CONT	R/W	0h	ADCINT4 Continue to Interrupt Mode 0 No further ADCINT4 pulses are generated until ADCINT4 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT4 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
13	INT4E	R/W	0h	ADCINT4 Interrupt Enable 0 ADCINT4 is disabled 1 ADCINT4 is enabled Reset type: SYSRSn
12	RESERVED	R	0h	Reserved
11-8	INT4SEL	R/W	0h	ADCINT4 EOC Source Select 0h EOC0 is trigger for ADCINT4 1h EOC1 is trigger for ADCINT4 2h EOC2 is trigger for ADCINT4 3h EOC3 is trigger for ADCINT4 4h EOC4 is trigger for ADCINT4 5h EOC5 is trigger for ADCINT4 6h EOC6 is trigger for ADCINT4 7h EOC7 is trigger for ADCINT4 8h EOC8 is trigger for ADCINT4 9h EOC9 is trigger for ADCINT4 Ah EOC10 is trigger for ADCINT4 Bh EOC11 is trigger for ADCINT4 Ch EOC12 is trigger for ADCINT4 Dh EOC13 is trigger for ADCINT4 Eh EOC14 is trigger for ADCINT4 Fh EOC15 is trigger for ADCINT4 Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	INT3CONT	R/W	0h	ADCINT3 Continue to Interrupt Mode 0 No further ADCINT3 pulses are generated until ADCINT3 flag (in ADCINTFLG register) is cleared by user. 1 ADCINT3 pulses are generated whenever an EOC pulse is generated irrespective of whether the flag bit is cleared or not. Reset type: SYSRSn
5	INT3E	R/W	0h	ADCINT3 Interrupt Enable 0 ADCINT3 is disabled 1 ADCINT3 is enabled Reset type: SYSRSn

**Table 11-44. ADCINTSEL3N4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R	0h	Reserved
3-0	INT3SEL	R/W	0h	ADCINT3 EOC Source Select 0h EOC0 is trigger for ADCINT3 1h EOC1 is trigger for ADCINT3 2h EOC2 is trigger for ADCINT3 3h EOC3 is trigger for ADCINT3 4h EOC4 is trigger for ADCINT3 5h EOC5 is trigger for ADCINT3 6h EOC6 is trigger for ADCINT3 7h EOC7 is trigger for ADCINT3 8h EOC8 is trigger for ADCINT3 9h EOC9 is trigger for ADCINT3 Ah EOC10 is trigger for ADCINT3 Bh EOC11 is trigger for ADCINT3 Ch EOC12 is trigger for ADCINT3 Dh EOC13 is trigger for ADCINT3 Eh EOC14 is trigger for ADCINT3 Fh EOC15 is trigger for ADCINT3 Reset type: SYSRStn

### 11.15.3.10 ADCSOCPRICTL Register (Offset = 9h) [Reset = 0200h]

ADCSOCPRICTL is shown in [Figure 11-50](#) and described in [Table 11-45](#).

Return to the [Summary Table](#).

ADC SOC Priority Control Register

**Figure 11-50. ADCSOCPRICTL Register**

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

**Table 11-45. ADCSOCPRICTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-5	RRPOINTER	R	10h	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. 00h SOC0 was last round robin SOC to convert, SOC1 is highest round robin priority. 01h SOC1 was last round robin SOC to convert, SOC2 is highest round robin priority. 02h SOC2 was last round robin SOC to convert, SOC3 is highest round robin priority. 03h SOC3 was last round robin SOC to convert, SOC4 is highest round robin priority. 04h SOC4 was last round robin SOC to convert, SOC5 is highest round robin priority. 05h SOC5 was last round robin SOC to convert, SOC6 is highest round robin priority. 06h SOC6 was last round robin SOC to convert, SOC7 is highest round robin priority. 07h SOC7 was last round robin SOC to convert, SOC8 is highest round robin priority. 08h SOC8 was last round robin SOC to convert, SOC9 is highest round robin priority. 09h SOC9 was last round robin SOC to convert, SOC10 is highest round robin priority. 0Ah SOC10 was last round robin SOC to convert, SOC11 is highest round robin priority. 0Bh SOC11 was last round robin SOC to convert, SOC12 is highest round robin priority. 0Ch SOC12 was last round robin SOC to convert, SOC13 is highest round robin priority. 0Dh SOC13 was last round robin SOC to convert, SOC14 is highest round robin priority. 0Eh SOC14 was last round robin SOC to convert, SOC15 is highest round robin priority. 0Fh SOC15 was last round robin SOC to convert, SOC0 is highest round robin priority. 10h Reset value to indicate no SOC has been converted. SOC0 is highest round robin priority. Set to this value when the ADC module is reset by SOFTPRES or when the ADCSOCPRICTL 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. Others Invalid value. Reset type: SYSRSn

**Table 11-45. ADCSOCPRCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	SOC PRIORITY	R/W	0h	<p>SOC Priority</p> <p>Determines the cutoff point for priority mode and round robin arbitration for SOCx</p> <p>00h SOC priority is handled in round robin mode for all channels.</p> <p>01h SOC0 is high priority, rest of channels are in round robin mode.</p> <p>02h SOC0-SOC1 are high priority, SOC2-SOC15 are in round robin mode.</p> <p>03h SOC0-SOC2 are high priority, SOC3-SOC15 are in round robin mode.</p> <p>04h SOC0-SOC3 are high priority, SOC4-SOC15 are in round robin mode.</p> <p>05h SOC0-SOC4 are high priority, SOC5-SOC15 are in round robin mode.</p> <p>06h SOC0-SOC5 are high priority, SOC6-SOC15 are in round robin mode.</p> <p>07h SOC0-SOC6 are high priority, SOC7-SOC15 are in round robin mode.</p> <p>08h SOC0-SOC7 are high priority, SOC8-SOC15 are in round robin mode.</p> <p>09h SOC0-SOC8 are high priority, SOC9-SOC15 are in round robin mode.</p> <p>0Ah SOC0-SOC9 are high priority, SOC10-SOC15 are in round robin mode.</p> <p>0Bh SOC0-SOC10 are high priority, SOC11-SOC15 are in round robin mode.</p> <p>0Ch SOC0-SOC11 are high priority, SOC12-SOC15 are in round robin mode.</p> <p>0Dh SOC0-SOC12 are high priority, SOC13-SOC15 are in round robin mode.</p> <p>0Eh SOC0-SOC13 are high priority, SOC14-SOC15 are in round robin mode.</p> <p>0Fh SOC0-SOC14 are high priority, SOC15 is in round robin mode.</p> <p>10h All SOCx are in high priority mode, arbitrated by SOC number.</p> <p>Others Invalid selection.</p> <p>Reset type: SYSRSn</p>

### 11.15.3.11 ADCINTSOCSEL1 Register (Offset = Ah) [Reset = 0000h]

ADCINTSOCSEL1 is shown in [Figure 11-51](#) and described in [Table 11-46](#).

Return to the [Summary Table](#).

ADC Interrupt SOC Selection 1 Register

**Figure 11-51. ADCINTSOCSEL1 Register**

15	14	13	12	11	10	9	8
SOC7		SOC6		SOC5		SOC4	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
SOC3		SOC2		SOC1		SOC0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 11-46. ADCINTSOCSEL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOC7	R/W	0h	SOC7 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC7. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC7. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC7. 10 ADCINT2 will trigger SOC7. 11 Invalid selection. Reset type: SYSRSn
13-12	SOC6	R/W	0h	SOC6 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC6. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC6. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC6. 10 ADCINT2 will trigger SOC6. 11 Invalid selection. Reset type: SYSRSn
11-10	SOC5	R/W	0h	SOC5 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC5. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC5. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC5. 10 ADCINT2 will trigger SOC5. 11 Invalid selection. Reset type: SYSRSn
9-8	SOC4	R/W	0h	SOC4 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC4. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC4. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC4. 10 ADCINT2 will trigger SOC4. 11 Invalid selection. Reset type: SYSRSn

**Table 11-46. ADCINTSOCSEL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	SOC3	R/W	0h	SOC3 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC3. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC3. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC3. 10 ADCINT2 will trigger SOC3. 11 Invalid selection. Reset type: SYSRSn
5-4	SOC2	R/W	0h	SOC2 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC2. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC2. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC2. 10 ADCINT2 will trigger SOC2. 11 Invalid selection. Reset type: SYSRSn
3-2	SOC1	R/W	0h	SOC1 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC1. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC1. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC1. 10 ADCINT2 will trigger SOC1. 11 Invalid selection. Reset type: SYSRSn
1-0	SOC0	R/W	0h	SOC0 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC0. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC0. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC0. 10 ADCINT2 will trigger SOC0. 11 Invalid selection. Reset type: SYSRSn

### 11.15.3.12 ADCINTSOCSEL2 Register (Offset = Bh) [Reset = 0000h]

ADCINTSOCSEL2 is shown in [Figure 11-52](#) and described in [Table 11-47](#).

Return to the [Summary Table](#).

ADC Interrupt SOC Selection 2 Register

**Figure 11-52. ADCINTSOCSEL2 Register**

15	14	13	12	11	10	9	8
SOC15		SOC14		SOC13		SOC12	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
SOC11		SOC10		SOC9		SOC8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 11-47. ADCINTSOCSEL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOC15	R/W	0h	SOC15 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC15. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC15. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC15. 10 ADCINT2 will trigger SOC15. 11 Invalid selection. Reset type: SYSRSn
13-12	SOC14	R/W	0h	SOC14 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC14. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC14. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC14. 10 ADCINT2 will trigger SOC14. 11 Invalid selection. Reset type: SYSRSn
11-10	SOC13	R/W	0h	SOC13 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC13. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC13. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC13. 10 ADCINT2 will trigger SOC13. 11 Invalid selection. Reset type: SYSRSn
9-8	SOC12	R/W	0h	SOC12 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC12. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC12. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC12. 10 ADCINT2 will trigger SOC12. 11 Invalid selection. Reset type: SYSRSn

**Table 11-47. ADCINTSOCSEL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-6	SOC11	R/W	0h	SOC11 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC11. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC11. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC11. 10 ADCINT2 will trigger SOC11. 11 Invalid selection. Reset type: SYSRSn
5-4	SOC10	R/W	0h	SOC10 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC10. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC10. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC10. 10 ADCINT2 will trigger SOC10. 11 Invalid selection. Reset type: SYSRSn
3-2	SOC9	R/W	0h	SOC9 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC9. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC9. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC9. 10 ADCINT2 will trigger SOC9. 11 Invalid selection. Reset type: SYSRSn
1-0	SOC8	R/W	0h	SOC8 ADC Interrupt Trigger Select. Selects which, if any, ADCINT triggers SOC8. The trigger selected in this field is in addition to the TRIGSEL field in the ADCSOCxCTL register. 00 No ADCINT will trigger SOC8. TRIGSEL field alone determines SOC0 trigger. 01 ADCINT1 will trigger SOC8. 10 ADCINT2 will trigger SOC8. 11 Invalid selection. Reset type: SYSRSn

### 11.15.3.13 ADCSOCFLG1 Register (Offset = Ch) [Reset = 0000h]

ADCSOCFLG1 is shown in [Figure 11-53](#) and described in [Table 11-48](#).

Return to the [Summary Table](#).

ADC SOC Flag 1 Register

**Figure 11-53. ADCSOCFLG1 Register**

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 11-48. 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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 11-48. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	SOC12	R	0h	<p>SOC12 Start of Conversion Flag. Indicates the state of SOC12 conversions.</p> <p>0 No sample pending for SOC12. 1 Trigger has been received and sample is pending for SOC12.</p> <p>This bit will be automatically cleared when the SOC12 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
11	SOC11	R	0h	<p>SOC11 Start of Conversion Flag. Indicates the state of SOC11 conversions.</p> <p>0 No sample pending for SOC11. 1 Trigger has been received and sample is pending for SOC11.</p> <p>This bit will be automatically cleared when the SOC11 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R	0h	<p>SOC10 Start of Conversion Flag. Indicates the state of SOC10 conversions.</p> <p>0 No sample pending for SOC10. 1 Trigger has been received and sample is pending for SOC10.</p> <p>This bit will be automatically cleared when the SOC10 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
9	SOC9	R	0h	<p>SOC9 Start of Conversion Flag. Indicates the state of SOC9 conversions.</p> <p>0 No sample pending for SOC9. 1 Trigger has been received and sample is pending for SOC9.</p> <p>This bit will be automatically cleared when the SOC9 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
8	SOC8	R	0h	<p>SOC8 Start of Conversion Flag. Indicates the state of SOC8 conversions.</p> <p>0 No sample pending for SOC8. 1 Trigger has been received and sample is pending for SOC8.</p> <p>This bit will be automatically cleared when the SOC8 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 11-48. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7	SOC7	R	0h	<p>SOC7 Start of Conversion Flag. Indicates the state of SOC7 conversions.</p> <p>0 No sample pending for SOC7. 1 Trigger has been received and sample is pending for SOC7.</p> <p>This bit will be automatically cleared when the SOC7 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R	0h	<p>SOC6 Start of Conversion Flag. Indicates the state of SOC6 conversions.</p> <p>0 No sample pending for SOC6. 1 Trigger has been received and sample is pending for SOC6.</p> <p>This bit will be automatically cleared when the SOC6 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
5	SOC5	R	0h	<p>SOC5 Start of Conversion Flag. Indicates the state of SOC5 conversions.</p> <p>0 No sample pending for SOC5. 1 Trigger has been received and sample is pending for SOC5.</p> <p>This bit will be automatically cleared when the SOC5 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
4	SOC4	R	0h	<p>SOC4 Start of Conversion Flag. Indicates the state of SOC4 conversions.</p> <p>0 No sample pending for SOC4. 1 Trigger has been received and sample is pending for SOC4.</p> <p>This bit will be automatically cleared when the SOC4 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R	0h	<p>SOC3 Start of Conversion Flag. Indicates the state of SOC3 conversions.</p> <p>0 No sample pending for SOC3. 1 Trigger has been received and sample is pending for SOC3.</p> <p>This bit will be automatically cleared when the SOC3 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 11-48. ADCSOCFLG1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SOC2	R	0h	<p>SOC2 Start of Conversion Flag. Indicates the state of SOC2 conversions.</p> <p>0 No sample pending for SOC2. 1 Trigger has been received and sample is pending for SOC2.</p> <p>This bit will be automatically cleared when the SOC2 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
1	SOC1	R	0h	<p>SOC1 Start of Conversion Flag. Indicates the state of SOC1 conversions.</p> <p>0 No sample pending for SOC1. 1 Trigger has been received and sample is pending for SOC1.</p> <p>This bit will be automatically cleared when the SOC1 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
0	SOC0	R	0h	<p>SOC0 Start of Conversion Flag. Indicates the state of SOC0 conversions.</p> <p>0 No sample pending for SOC0. 1 Trigger has been received and sample is pending for SOC0.</p> <p>This bit will be automatically cleared when the SOC0 conversion is started. If contention exists where this bit receives both a request to set and a request to clear on the same cycle, regardless of the source of either, this bit will be set and the request to clear will be ignored. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether this bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

### 11.15.3.14 ADCSOCFRC1 Register (Offset = Dh) [Reset = 0000h]

ADCSOCFRC1 is shown in [Figure 11-54](#) and described in [Table 11-49](#).

Return to the [Summary Table](#).

ADC SOC Force 1 Register

**Figure 11-54. ADCSOCFRC1 Register**

15		14		13		12		11		10		9		8	
SOC15		SOC14		SOC13		SOC12		SOC11		SOC10		SOC9		SOC8	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	
7		6		5		4		3		2		1		0	
SOC7		SOC6		SOC5		SOC4		SOC3		SOC2		SOC1		SOC0	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	

**Table 11-49. ADCSOCFRC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R-0/W1S	0h	<p>SOC15 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC15 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC15 flag bit to 1. This will cause a conversion to start once priority is given to SOC15.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC15 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
14	SOC14	R-0/W1S	0h	<p>SOC14 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC14 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC14 flag bit to 1. This will cause a conversion to start once priority is given to SOC14.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC14 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
13	SOC13	R-0/W1S	0h	<p>SOC13 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC13 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC13 flag bit to 1. This will cause a conversion to start once priority is given to SOC13.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC13 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 11-49. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12	SOC12	R-0/W1S	0h	<p>SOC12 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC12 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC12 flag bit to 1. This will cause a conversion to start once priority is given to SOC12.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC12 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
11	SOC11	R-0/W1S	0h	<p>SOC11 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC11 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC11 flag bit to 1. This will cause a conversion to start once priority is given to SOC11.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC11 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R-0/W1S	0h	<p>SOC10 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC10 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC10 flag bit to 1. This will cause a conversion to start once priority is given to SOC10.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC10 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
9	SOC9	R-0/W1S	0h	<p>SOC9 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC9 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC9 flag bit to 1. This will cause a conversion to start once priority is given to SOC9.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC9 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 11-49. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	SOC8	R-0/W1S	0h	<p>SOC8 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC8 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC8 flag bit to 1. This will cause a conversion to start once priority is given to SOC8.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC8 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
7	SOC7	R-0/W1S	0h	<p>SOC7 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC7 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC7 flag bit to 1. This will cause a conversion to start once priority is given to SOC7.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC7 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R-0/W1S	0h	<p>SOC6 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC6 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC6 flag bit to 1. This will cause a conversion to start once priority is given to SOC6.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC6 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
5	SOC5	R-0/W1S	0h	<p>SOC5 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC5 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC5 flag bit to 1. This will cause a conversion to start once priority is given to SOC5.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC5 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>

**Table 11-49. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SOC4	R-0/W1S	0h	<p>SOC4 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC4 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC4 flag bit to 1. This will cause a conversion to start once priority is given to SOC4.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC4 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R-0/W1S	0h	<p>SOC3 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC3 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC3 flag bit to 1. This will cause a conversion to start once priority is given to SOC3.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC3 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
2	SOC2	R-0/W1S	0h	<p>SOC2 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC2 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</p> <p>1 Force SOC2 flag bit to 1. This will cause a conversion to start once priority is given to SOC2.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to clear the SOC2 bit in the ADCSOCFLG1 register, then software has priority and the ADCSOCFLG1 bit will be set. In this case the overflow bit in the ADCSOCOVF1 register will not be affected regardless of whether the ADCSOCFLG1 bit was previously set or not.</p> <p>Reset type: SYSRSn</p>
1	SOC1	R-0/W1S	0h	<p>SOC1 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC1 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</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> <p>Reset type: SYSRSn</p>

**Table 11-49. ADCSOCFRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	SOC0	R-0/W1S	0h	<p>SOC0 Force Start of Conversion Bit. Writing a 1 will force to 1 the SOC0 flag in the ADCSOCFLG1 register. This can be used to initiate a software initiated conversion. Writes of 0 are ignored. This bit will always read as a 0.</p> <p>0 No action.</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> <p>Reset type: SYSRSn</p>

### 11.15.3.15 ADCSOCOVF1 Register (Offset = Eh) [Reset = 0000h]

ADCSOCOVF1 is shown in [Figure 11-55](#) and described in [Table 11-50](#).

Return to the [Summary Table](#).

ADC SOC Overflow 1 Register

**Figure 11-55. ADCSOCOVF1 Register**

15	14	13	12	11	10	9	8
SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 11-50. 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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 11-50. 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. 1 SOC11 event overflow.</p> <p>An overflow condition does not stop SOC11 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
10	SOC10	R	0h	<p>SOC10 Start of Conversion Overflow Flag. Indicates an SOC10 event was generated in hardware while an existing SOC10 event was already pending.</p> <p>0 No SOC10 event overflow. 1 SOC10 event overflow.</p> <p>An overflow condition does not stop SOC10 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
9	SOC9	R	0h	<p>SOC9 Start of Conversion Overflow Flag. Indicates an SOC9 event was generated in hardware while an existing SOC9 event was already pending.</p> <p>0 No SOC9 event overflow. 1 SOC9 event overflow.</p> <p>An overflow condition does not stop SOC9 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
8	SOC8	R	0h	<p>SOC8 Start of Conversion Overflow Flag. Indicates an SOC8 event was generated in hardware while an existing SOC8 event was already pending.</p> <p>0 No SOC8 event overflow. 1 SOC8 event overflow.</p> <p>An overflow condition does not stop SOC8 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
7	SOC7	R	0h	<p>SOC7 Start of Conversion Overflow Flag. Indicates an SOC7 event was generated in hardware while an existing SOC7 event was already pending.</p> <p>0 No SOC7 event overflow. 1 SOC7 event overflow.</p> <p>An overflow condition does not stop SOC7 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
6	SOC6	R	0h	<p>SOC6 Start of Conversion Overflow Flag. Indicates an SOC6 event was generated in hardware while an existing SOC6 event was already pending.</p> <p>0 No SOC6 event overflow. 1 SOC6 event overflow.</p> <p>An overflow condition does not stop SOC6 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

**Table 11-50. 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. 1 SOC5 event overflow.</p> <p>An overflow condition does not stop SOC5 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
4	SOC4	R	0h	<p>SOC4 Start of Conversion Overflow Flag. Indicates an SOC4 event was generated in hardware while an existing SOC4 event was already pending.</p> <p>0 No SOC4 event overflow. 1 SOC4 event overflow.</p> <p>An overflow condition does not stop SOC4 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
3	SOC3	R	0h	<p>SOC3 Start of Conversion Overflow Flag. Indicates an SOC3 event was generated in hardware while an existing SOC3 event was already pending.</p> <p>0 No SOC3 event overflow. 1 SOC3 event overflow.</p> <p>An overflow condition does not stop SOC3 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
2	SOC2	R	0h	<p>SOC2 Start of Conversion Overflow Flag. Indicates an SOC2 event was generated in hardware while an existing SOC2 event was already pending.</p> <p>0 No SOC2 event overflow. 1 SOC2 event overflow.</p> <p>An overflow condition does not stop SOC2 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
1	SOC1	R	0h	<p>SOC1 Start of Conversion Overflow Flag. Indicates an SOC1 event was generated in hardware while an existing SOC1 event was already pending.</p> <p>0 No SOC1 event overflow. 1 SOC1 event overflow.</p> <p>An overflow condition does not stop SOC1 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>
0	SOC0	R	0h	<p>SOC0 Start of Conversion Overflow Flag. Indicates an SOC0 event was generated in hardware while an existing SOC0 event was already pending.</p> <p>0 No SOC0 event overflow. 1 SOC0 event overflow.</p> <p>An overflow condition does not stop SOC0 events from being processed. It simply is an indication that a hardware trigger was missed. A write to the ADCSOCFRC1 register does not affect this bit.</p> <p>Reset type: SYSRSn</p>

### 11.15.3.16 ADCSOCOVFCLR1 Register (Offset = Fh) [Reset = 0000h]

ADCSOCOVFCLR1 is shown in [Figure 11-56](#) and described in [Table 11-51](#).

Return to the [Summary Table](#).

ADC SOC Overflow Clear 1 Register

**Figure 11-56. ADCSOCOVFCLR1 Register**

15		14		13		12		11		10		9		8	
SOC15		SOC14		SOC13		SOC12		SOC11		SOC10		SOC9		SOC8	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	
7		6		5		4		3		2		1		0	
SOC7		SOC6		SOC5		SOC4		SOC3		SOC2		SOC1		SOC0	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h	

**Table 11-51. ADCSOCOVFCLR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOC15	R-0/W1S	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.. Reset type: SYSRSn
14	SOC14	R-0/W1S	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.. Reset type: SYSRSn
13	SOC13	R-0/W1S	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.. Reset type: SYSRSn
12	SOC12	R-0/W1S	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.. Reset type: SYSRSn

**Table 11-51. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	SOC11	R-0/W1S	0h	<p>SOC11 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC11 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC11 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
10	SOC10	R-0/W1S	0h	<p>SOC10 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC10 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC10 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
9	SOC9	R-0/W1S	0h	<p>SOC9 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC9 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC9 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
8	SOC8	R-0/W1S	0h	<p>SOC8 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC8 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC8 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
7	SOC7	R-0/W1S	0h	<p>SOC7 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC7 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC7 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
6	SOC6	R-0/W1S	0h	<p>SOC6 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC6 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC6 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>

**Table 11-51. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SOC5	R-0/W1S	0h	<p>SOC5 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC5 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC5 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
4	SOC4	R-0/W1S	0h	<p>SOC4 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC4 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC4 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
3	SOC3	R-0/W1S	0h	<p>SOC3 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC3 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC3 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
2	SOC2	R-0/W1S	0h	<p>SOC2 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC2 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC2 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
1	SOC1	R-0/W1S	0h	<p>SOC1 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC1 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC1 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>
0	SOC0	R-0/W1S	0h	<p>SOC0 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC0 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.</p> <p>0 No action. 1 Clear SOC0 overflow flag.</p> <p>If software tries to set this bit on the same clock cycle that hardware tries to set the overflow bit in the ADCSOCOVF1 register, then hardware has priority and the ADCSOCOVF1 bit will be set..</p> <p>Reset type: SYSRSn</p>

### 11.15.3.17 ADCSOC0CTL Register (Offset = 10h) [Reset = 0000000h]

ADCSOC0CTL is shown in [Figure 11-57](#) and described in [Table 11-52](#).

Return to the [Summary Table](#).

ADC SOC0 Control Register

**Figure 11-57. ADCSOC0CTL Register**

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

**Table 11-52. ADCSOC0CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC0 Trigger Source Select. Along with the SOC0 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC0 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC0s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC0 Channel Select. Selects the channel to be converted when SOC0 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-52. ADCSOC0CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC0 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.18 ADCSOC1CTL Register (Offset = 12h) [Reset = 0000000h]

ADCSOC1CTL is shown in [Figure 11-58](#) and described in [Table 11-53](#).

Return to the [Summary Table](#).

ADC SOC1 Control Register

**Figure 11-58. ADCSOC1CTL Register**

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

**Table 11-53. ADCSOC1CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	SOC1 Trigger Source Select. Along with the SOC1 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC1 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. Note: SOCFRC1 register can always be used to software trigger SOC1s in addition to any hardware trigger configuration. 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, Input X-Bar INPUT5 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h - 3Fh - Reserved Reset type: SYSRSn
19-15	CHSEL	R/W	0h	SOC1 Channel Select. Selects the channel to be converted when SOC1 is received by the ADC. 00h ADCIN0 01h ADCIN1 02h ADCIN2 03h ADCIN3 ... 12h ADCIN18 13h ADCIN19 14h ADCIN20 15h - 1Fh Reserved Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved

**Table 11-53. ADCSOC1CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC1 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.19 ADCSOC2CTL Register (Offset = 14h) [Reset = 0000000h]

ADCSOC2CTL is shown in [Figure 11-59](#) and described in [Table 11-54](#).

Return to the [Summary Table](#).

ADC SOC2 Control Register

**Figure 11-59. ADCSOC2CTL Register**

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

**Table 11-54. ADCSOC2CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC2 Trigger Source Select. Along with the SOC2 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC2 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC2s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC2 Channel Select. Selects the channel to be converted when SOC2 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-54. ADCSOC2CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC2 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.20 ADCSOC3CTL Register (Offset = 16h) [Reset = 0000000h]

ADCSOC3CTL is shown in [Figure 11-60](#) and described in [Table 11-55](#).

Return to the [Summary Table](#).

ADC SOC3 Control Register

**Figure 11-60. ADCSOC3CTL Register**

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

**Table 11-55. ADCSOC3CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC3 Trigger Source Select. Along with the SOC3 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC3 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC3s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC3 Channel Select. Selects the channel to be converted when SOC3 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-55. ADCSOC3CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC3 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.21 ADCSOC4CTL Register (Offset = 18h) [Reset = 0000000h]

ADCSOC4CTL is shown in [Figure 11-61](#) and described in [Table 11-56](#).

Return to the [Summary Table](#).

ADC SOC4 Control Register

**Figure 11-61. ADCSOC4CTL Register**

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

**Table 11-56. ADCSOC4CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	SOC4 Trigger Source Select. Along with the SOC4 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC4 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. Note: SOCFRC1 register can always be used to software trigger SOC4s in addition to any hardware trigger configuration. 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, Input X-Bar INPUT5 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h - 3Fh - Reserved Reset type: SYSRSn
19-15	CHSEL	R/W	0h	SOC4 Channel Select. Selects the channel to be converted when SOC4 is received by the ADC. 00h ADCIN0 01h ADCIN1 02h ADCIN2 03h ADCIN3 ... 12h ADCIN18 13h ADCIN19 14h ADCIN20 15h - 1Fh Reserved Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved

**Table 11-56. ADCSOC4CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC4 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.22 ADCSOC5CTL Register (Offset = 1Ah) [Reset = 0000000h]

ADCSOC5CTL is shown in [Figure 11-62](#) and described in [Table 11-57](#).

Return to the [Summary Table](#).

ADC SOC5 Control Register

**Figure 11-62. ADCSOC5CTL Register**

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

**Table 11-57. ADCSOC5CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC5 Trigger Source Select. Along with the SOC5 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC5 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC5s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC5 Channel Select. Selects the channel to be converted when SOC5 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-57. ADCSOC5CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC5 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.23 ADCSOC6CTL Register (Offset = 1Ch) [Reset = 0000000h]

ADCSOC6CTL is shown in [Figure 11-63](#) and described in [Table 11-58](#).

Return to the [Summary Table](#).

ADC SOC6 Control Register

**Figure 11-63. ADCSOC6CTL Register**

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

**Table 11-58. ADCSOC6CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	SOC6 Trigger Source Select. Along with the SOC6 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC6 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. Note: SOCFRC1 register can always be used to software trigger SOC6s in addition to any hardware trigger configuration. 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, Input X-Bar INPUT5 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h - 3Fh - Reserved Reset type: SYSRSn
19-15	CHSEL	R/W	0h	SOC6 Channel Select. Selects the channel to be converted when SOC6 is received by the ADC. 00h ADCIN0 01h ADCIN1 02h ADCIN2 03h ADCIN3 ... 12h ADCIN18 13h ADCIN19 14h ADCIN20 15h - 1Fh Reserved Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved

**Table 11-58. ADCSOC6CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC6 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.24 ADCSOC7CTL Register (Offset = 1Eh) [Reset = 0000000h]

ADCSOC7CTL is shown in [Figure 11-64](#) and described in [Table 11-59](#).

Return to the [Summary Table](#).

ADC SOC7 Control Register

**Figure 11-64. ADCSOC7CTL Register**

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

**Table 11-59. ADCSOC7CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC7 Trigger Source Select. Along with the SOC7 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC7 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC7s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC7 Channel Select. Selects the channel to be converted when SOC7 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-59. ADCSOC7CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC7 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.25 ADCSOC8CTL Register (Offset = 20h) [Reset = 0000000h]

ADCSOC8CTL is shown in [Figure 11-65](#) and described in [Table 11-60](#).

Return to the [Summary Table](#).

ADC SOC8 Control Register

**Figure 11-65. ADCSOC8CTL Register**

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

**Table 11-60. ADCSOC8CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	SOC8 Trigger Source Select. Along with the SOC8 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC8 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it. Note: SOCFRC1 register can always be used to software trigger SOC8s in addition to any hardware trigger configuration. 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, Input X-Bar INPUT5 05h ADCTRIG5 - ePWM1, ADCSOCA 06h ADCTRIG6 - ePWM1, ADCSOCB 07h ADCTRIG7 - ePWM2, ADCSOCA 08h ADCTRIG8 - ePWM2, ADCSOCB 09h ADCTRIG9 - ePWM3, ADCSOCA 0Ah ADCTRIG10 - ePWM3, ADCSOCB 0Bh ADCTRIG11 - ePWM4, ADCSOCA 0Ch ADCTRIG12 - ePWM4, ADCSOCB 0Dh ADCTRIG13 - ePWM5, ADCSOCA 0Eh ADCTRIG14 - ePWM5, ADCSOCB 0Fh ADCTRIG15 - ePWM6, ADCSOCA 10h ADCTRIG16 - ePWM6, ADCSOCB 11h ADCTRIG17 - ePWM7, ADCSOCA 12h ADCTRIG18 - ePWM7, ADCSOCB 13h - 3Fh - Reserved Reset type: SYSRSn
19-15	CHSEL	R/W	0h	SOC8 Channel Select. Selects the channel to be converted when SOC8 is received by the ADC. 00h ADCIN0 01h ADCIN1 02h ADCIN2 03h ADCIN3 ... 12h ADCIN18 13h ADCIN19 14h ADCIN20 15h - 1Fh Reserved Reset type: SYSRSn
14-9	RESERVED	R	0h	Reserved

**Table 11-60. ADCSOC8CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC8 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.26 ADCSOC9CTL Register (Offset = 22h) [Reset = 0000000h]

ADCSOC9CTL is shown in [Figure 11-66](#) and described in [Table 11-61](#).

Return to the [Summary Table](#).

ADC SOC9 Control Register

**Figure 11-66. ADCSOC9CTL Register**

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

**Table 11-61. ADCSOC9CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC9 Trigger Source Select. Along with the SOC9 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC9 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC9s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5  05h ADCTRIG5 - ePWM1, ADCSOCA  06h ADCTRIG6 - ePWM1, ADCSOCB  07h ADCTRIG7 - ePWM2, ADCSOCA  08h ADCTRIG8 - ePWM2, ADCSOCB  09h ADCTRIG9 - ePWM3, ADCSOCA  0Ah ADCTRIG10 - ePWM3, ADCSOCB  0Bh ADCTRIG11 - ePWM4, ADCSOCA  0Ch ADCTRIG12 - ePWM4, ADCSOCB  0Dh ADCTRIG13 - ePWM5, ADCSOCA  0Eh ADCTRIG14 - ePWM5, ADCSOCB  0Fh ADCTRIG15 - ePWM6, ADCSOCA  10h ADCTRIG16 - ePWM6, ADCSOCB  11h ADCTRIG17 - ePWM7, ADCSOCA  12h ADCTRIG18 - ePWM7, ADCSOCB  13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC9 Channel Select. Selects the channel to be converted when SOC9 is received by the ADC.</p> <p>00h ADCIN0  01h ADCIN1  02h ADCIN2  03h ADCIN3  ...  12h ADCIN18  13h ADCIN19  14h ADCIN20  15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-61. ADCSOC9CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC9 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.27 ADCSOC10CTL Register (Offset = 24h) [Reset = 0000000h]

ADCSOC10CTL is shown in [Figure 11-67](#) and described in [Table 11-62](#).

Return to the [Summary Table](#).

ADC SOC10 Control Register

**Figure 11-67. ADCSOC10CTL Register**

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

**Table 11-62. ADCSOC10CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC10 Trigger Source Select. Along with the SOC10 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC10 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC's in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC10 Channel Select. Selects the channel to be converted when SOC10 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-62. ADCSOC10CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC10 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.28 ADCSOC11CTL Register (Offset = 26h) [Reset = 0000000h]

ADCSOC11CTL is shown in [Figure 11-68](#) and described in [Table 11-63](#).

Return to the [Summary Table](#).

ADC SOC11 Control Register

**Figure 11-68. ADCSOC11CTL Register**

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

**Table 11-63. ADCSOC11CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC11 Trigger Source Select. Along with the SOC11 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC11 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC's in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC11 Channel Select. Selects the channel to be converted when SOC11 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-63. ADCSOC11CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC11 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.29 ADCSOC12CTL Register (Offset = 28h) [Reset = 0000000h]

ADCSOC12CTL is shown in [Figure 11-69](#) and described in [Table 11-64](#).

Return to the [Summary Table](#).

ADC SOC12 Control Register

**Figure 11-69. ADCSOC12CTL Register**

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

**Table 11-64. ADCSOC12CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC12 Trigger Source Select. Along with the SOC12 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC12 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC12s in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC12 Channel Select. Selects the channel to be converted when SOC12 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-64. ADCSOC12CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC12 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.30 ADCSOC13CTL Register (Offset = 2Ah) [Reset = 0000000h]

ADCSOC13CTL is shown in [Figure 11-70](#) and described in [Table 11-65](#).

Return to the [Summary Table](#).

ADC SOC13 Control Register

**Figure 11-70. ADCSOC13CTL Register**

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

**Table 11-65. ADCSOC13CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC13 Trigger Source Select. Along with the SOC13 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC13 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC's in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5  05h ADCTRIG5 - ePWM1, ADCSOCA  06h ADCTRIG6 - ePWM1, ADCSOCB  07h ADCTRIG7 - ePWM2, ADCSOCA  08h ADCTRIG8 - ePWM2, ADCSOCB  09h ADCTRIG9 - ePWM3, ADCSOCA  0Ah ADCTRIG10 - ePWM3, ADCSOCB  0Bh ADCTRIG11 - ePWM4, ADCSOCA  0Ch ADCTRIG12 - ePWM4, ADCSOCB  0Dh ADCTRIG13 - ePWM5, ADCSOCA  0Eh ADCTRIG14 - ePWM5, ADCSOCB  0Fh ADCTRIG15 - ePWM6, ADCSOCA  10h ADCTRIG16 - ePWM6, ADCSOCB  11h ADCTRIG17 - ePWM7, ADCSOCA  12h ADCTRIG18 - ePWM7, ADCSOCB  13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC13 Channel Select. Selects the channel to be converted when SOC13 is received by the ADC.</p> <p>00h ADCIN0  01h ADCIN1  02h ADCIN2  03h ADCIN3  ...  12h ADCIN18  13h ADCIN19  14h ADCIN20  15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-65. ADCSOC13CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC13 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.31 ADCSOC14CTL Register (Offset = 2Ch) [Reset = 0000000h]

ADCSOC14CTL is shown in [Figure 11-71](#) and described in [Table 11-66](#).

Return to the [Summary Table](#).

ADC SOC14 Control Register

**Figure 11-71. ADCSOC14CTL Register**

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

**Table 11-66. ADCSOC14CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC14 Trigger Source Select. Along with the SOC14 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC14 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC's in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC14 Channel Select. Selects the channel to be converted when SOC14 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-66. ADCSOC14CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC14 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.32 ADCSOC15CTL Register (Offset = 2Eh) [Reset = 0000000h]

ADCSOC15CTL is shown in [Figure 11-72](#) and described in [Table 11-67](#).

Return to the [Summary Table](#).

ADC SOC15 Control Register

**Figure 11-72. ADCSOC15CTL Register**

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

**Table 11-67. ADCSOC15CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reserved
26-20	TRIGSEL	R/W	0h	<p>SOC15 Trigger Source Select. Along with the SOC15 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC15 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.</p> <p>Note: SOCFRC1 register can always be used to software trigger SOC15 in addition to any hardware trigger configuration.</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, Input X-Bar INPUT5            05h ADCTRIG5 - ePWM1, ADCSOCA            06h ADCTRIG6 - ePWM1, ADCSOCB            07h ADCTRIG7 - ePWM2, ADCSOCA            08h ADCTRIG8 - ePWM2, ADCSOCB            09h ADCTRIG9 - ePWM3, ADCSOCA            0Ah ADCTRIG10 - ePWM3, ADCSOCB            0Bh ADCTRIG11 - ePWM4, ADCSOCA            0Ch ADCTRIG12 - ePWM4, ADCSOCB            0Dh ADCTRIG13 - ePWM5, ADCSOCA            0Eh ADCTRIG14 - ePWM5, ADCSOCB            0Fh ADCTRIG15 - ePWM6, ADCSOCA            10h ADCTRIG16 - ePWM6, ADCSOCB            11h ADCTRIG17 - ePWM7, ADCSOCA            12h ADCTRIG18 - ePWM7, ADCSOCB            13h - 3Fh - Reserved</p> <p>Reset type: SYSRSn</p>
19-15	CHSEL	R/W	0h	<p>SOC15 Channel Select. Selects the channel to be converted when SOC15 is received by the ADC.</p> <p>00h ADCIN0            01h ADCIN1            02h ADCIN2            03h ADCIN3            ...            12h ADCIN18            13h ADCIN19            14h ADCIN20            15h - 1Fh Reserved</p> <p>Reset type: SYSRSn</p>
14-9	RESERVED	R	0h	Reserved

**Table 11-67. ADCSOC15CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-0	ACQPS	R/W	0h	SOC15 Acquisition Prescale. Controls the sample and hold window for this SOC. The configured acquisition time must be at least as long as one ADCCLK cycle for correct ADC operation. The device datasheet will also specify a minimum sample and hold window duration. 000h Sample window is 1 system clock cycle wide 001h Sample window is 2 system clock cycles wide 002h Sample window is 3 system clock cycles wide ... 1FFh Sample window is 512 system clock cycles wide Reset type: SYSRSn

### 11.15.3.33 ADCEVTSTAT Register (Offset = 30h) [Reset = 0000h]

ADCEVTSTAT is shown in [Figure 11-73](#) and described in [Table 11-68](#).

Return to the [Summary Table](#).

ADC Event Status Register

**Figure 11-73. 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

**Table 11-68. ADCEVTSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R	0h	Post Processing Block 4 Zero Crossing Flag. When set indicates the ADCPPB4RESULT register has changed sign. This bit is gated by EOC signal. Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
13	PPB4TRIPLO	R	0h	Post Processing Block 4 Trip Low Flag. When set indicates a digital compare trip low event has occurred. Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
12	PPB4TRIPHI	R	0h	Post Processing Block 4 Trip High Flag. When set indicates a digital compare trip high event has occurred. Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R	0h	Post Processing Block 3 Zero Crossing Flag. When set indicates the ADCPPB3RESULT register has changed sign. This bit is gated by EOC signal. Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn

**Table 11-68. ADCEVTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	PPB3TRIPLO	R	0h	<p>Post Processing Block 3 Trip Low Flag. When set indicates a digital compare trip low event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
8	PPB3TRIPHI	R	0h	<p>Post Processing Block 3 Trip High Flag. When set indicates a digital compare trip high event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R	0h	<p>Post Processing Block 2 Zero Crossing Flag. When set indicates the ADCPPB2RESULT register has changed sign. This bit is gated by EOC signal.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
5	PPB2TRIPLO	R	0h	<p>Post Processing Block 2 Trip Low Flag. When set indicates a digital compare trip low event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
4	PPB2TRIPHI	R	0h	<p>Post Processing Block 2 Trip High Flag. When set indicates a digital compare trip high event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R	0h	<p>Post Processing Block 1 Zero Crossing Flag. When set indicates the ADCPPB1RESULT register has changed sign. This bit is gated by EOC signal.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>

**Table 11-68. ADCEVTSTAT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	PPB1TRIPLO	R	0h	<p>Post Processing Block 1 Trip Low Flag. When set indicates a digital compare trip low event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>
0	PPB1TRIPHI	R	0h	<p>Post Processing Block 1 Trip High Flag. When set indicates a digital compare trip high event has occurred.</p> <p>Note: these bits are set even when the corresponding enable in ADCEVTINTSEL is not set. Because of this, an ISR may need to examine both the ADCEVTSTAT and ADCEVTINTSEL registers to determine the source of the interrupt.</p> <p>Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority</p> <p>Reset type: SYSRSn</p>

### 11.15.3.34 ADCEVTCLR Register (Offset = 32h) [Reset = 0000h]

ADCEVTCLR is shown in [Figure 11-74](#) and described in [Table 11-69](#).

Return to the [Summary Table](#).

ADC Event Clear Register

**Figure 11-74. ADCEVTCLR Register**

15	14	13	12	11	10	9	8
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI
R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h

**Table 11-69. ADCEVTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R-0/W1S	0h	Post Processing Block 4 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
13	PPB4TRIPLO	R-0/W1S	0h	Post Processing Block 4 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
12	PPB4TRIPHI	R-0/W1S	0h	Post Processing Block 4 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R-0/W1S	0h	Post Processing Block 3 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
9	PPB3TRIPLO	R-0/W1S	0h	Post Processing Block 3 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
8	PPB3TRIPHI	R-0/W1S	0h	Post Processing Block 3 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R-0/W1S	0h	Post Processing Block 2 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn

**Table 11-69. ADCEVTCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	PPB2TRIPLO	R-0/W1S	0h	Post Processing Block 2 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
4	PPB2TRIPHI	R-0/W1S	0h	Post Processing Block 2 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R-0/W1S	0h	Post Processing Block 1 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
1	PPB1TRIPLO	R-0/W1S	0h	Post Processing Block 1 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn
0	PPB1TRIPHI	R-0/W1S	0h	Post Processing Block 1 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register. Note: If software sets the clear bit on the same cycle that hardware is trying to set the flag bit, then hardware has priority Reset type: SYSRSn

### 11.15.3.35 ADCEVTSEL Register (Offset = 34h) [Reset = 0000h]

ADCEVTSEL is shown in [Figure 11-75](#) and described in [Table 11-70](#).

Return to the [Summary Table](#).

ADC Event Selection Register

**Figure 11-75. ADCEVTSEL Register**

15	14	13	12	11	10	9	8
RESERVED	PPB4ZERO	PPB4TRIPLO	PPB4TRIPHI	RESERVED	PPB3ZERO	PPB3TRIPLO	PPB3TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	PPB2ZERO	PPB2TRIPLO	PPB2TRIPHI	RESERVED	PPB1ZERO	PPB1TRIPLO	PPB1TRIPHI
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

**Table 11-70. ADCEVTSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R/W	0h	Post Processing Block 4 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R/W	0h	Post Processing Block 3 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R/W	0h	Post Processing Block 2 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn

**Table 11-70. ADCEVTSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R/W	0h	Post Processing Block 1 Zero Crossing Event Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Event Enable. Setting this bit allows the corresponding rising trip low flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Event Enable. Setting this bit allows the corresponding rising trip high flag to activate the event signal to the PWM blocks. The flag must be cleared before it can produce additional events to the PWM blocks. Reset type: SYSRSn

### 11.15.3.36 ADCEVTINTSEL Register (Offset = 36h) [Reset = 0000h]

ADCEVTINTSEL is shown in [Figure 11-76](#) and described in [Table 11-71](#).

Return to the [Summary Table](#).

ADC Event Interrupt Selection Register

**Figure 11-76. 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

**Table 11-71. ADCEVTINTSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	PPB4ZERO	R/W	0h	Post Processing Block 4 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
13	PPB4TRIPLO	R/W	0h	Post Processing Block 4 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
12	PPB4TRIPHI	R/W	0h	Post Processing Block 4 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
11	RESERVED	R	0h	Reserved
10	PPB3ZERO	R/W	0h	Post Processing Block 3 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
9	PPB3TRIPLO	R/W	0h	Post Processing Block 3 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
8	PPB3TRIPHI	R/W	0h	Post Processing Block 3 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
7	RESERVED	R	0h	Reserved
6	PPB2ZERO	R/W	0h	Post Processing Block 2 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn

**Table 11-71. ADCEVTINTSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	PPB2TRIPLO	R/W	0h	Post Processing Block 2 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
4	PPB2TRIPHI	R/W	0h	Post Processing Block 2 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
3	RESERVED	R	0h	Reserved
2	PPB1ZERO	R/W	0h	Post Processing Block 1 Zero Crossing Interrupt Enable. Setting this bit allows the corresponding rising zero crossing flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
1	PPB1TRIPLO	R/W	0h	Post Processing Block 1 Trip Low Interrupt Enable. Setting this bit allows the corresponding rising trip low flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn
0	PPB1TRIPHI	R/W	0h	Post Processing Block 1 Trip High Interrupt Enable. Setting this bit allows the corresponding rising trip high flag to activate the event interrupt signal to the PIE. The flag must be cleared before it can produce additional interrupts to the PIE. Reset type: SYSRSn

### 11.15.3.37 ADCOSDETECT Register (Offset = 38h) [Reset = 0000h]

ADCOSDETECT is shown in [Figure 11-77](#) and described in [Table 11-72](#).

Return to the [Summary Table](#).

ADC Open and Shorts Detect Register

**Figure 11-77. ADCOSDETECT Register**

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

**Table 11-72. ADCOSDETECT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2-0	DETECTCFG	R/W	0h	ADC Opens and Shorts Detect Configuration. This bit field defines the open/shorts detection circuit state. 0h Open/Shorts detection circuit is disabled. 1h Open/Shorts detection circuit is enabled at zero scale. 2h Open/Shorts detection circuit is enabled at full scale. 3h Open/Shorts detection circuit is enabled at (nominal) 5/12 scale. 4h Open/Shorts detection circuit is enabled at (nominal) 7/12 scale. 5h Open/Shorts detection circuit is enabled with a (nominal) 5K pulldown to VSSA. 6h Open/Shorts detection circuit is enabled with a (nominal) 5K pullup to VDDA. 7h Open/Shorts detection circuit is enabled with a (nominal) 7K pulldown to VSSA. Reset type: SYSRSn

### 11.15.3.38 ADCCOUNTER Register (Offset = 39h) [Reset = 0000h]

ADCCOUNTER is shown in [Figure 11-78](#) and described in [Table 11-73](#).

Return to the [Summary Table](#).

ADC Counter Register

**Figure 11-78. ADCCOUNTER Register**

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

**Table 11-73. ADCCOUNTER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	FREECOUNT	R	0h	ADC Free Running Counter Value. This bit field reflects the status of the free running ADC counter. Reset type: SYSRSn

### 11.15.3.39 ADCREV Register (Offset = 3Ah) [Reset = 0105h]

ADCREV is shown in [Figure 11-79](#) and described in [Table 11-74](#).

Return to the [Summary Table](#).

ADC Revision Register

**Figure 11-79. ADCREV Register**

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

**Table 11-74. ADCREV Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	REV	R	1h	ADC Revision. To allow documentation of differences between revisions. First version is labeled as 00h. Reset type: SYSRSn
7-0	TYPE	R	5h	ADC Type. Always set to 5 for this ADC. Reset type: SYSRSn

### 11.15.3.40 ADCOFFTRIM Register (Offset = 3Bh) [Reset = 0000h]

ADCOFFTRIM is shown in [Figure 11-80](#) and described in [Table 11-75](#).

Return to the [Summary Table](#).

ADC Offset Trim Register

**Figure 11-80. ADCOFFTRIM Register**

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

**Table 11-75. ADCOFFTRIM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R/W	0h	Reserved
7-0	OFFTRIM	R/W	0h	<p>ADC Offset Trim</p> <p>Adjusts the conversion results of the converter up or down to account for offset error in the ADC. A factory trim setting will be loaded during device boot.</p> <p>Offset can be corrected in the range of +7 to -8 LSBs. Value is <math>16 \times \text{Offset}</math> in 8-bit 2's complement:</p> <p>7 LSB (<math>16 \times 7</math>) = 112</p> <p>6 LSB (<math>16 \times 6</math>) = 96</p> <p>5 LSB (<math>16 \times 5</math>) = 80</p> <p>4 LSB (<math>16 \times 4</math>) = 64</p> <p>3 LSB (<math>16 \times 3</math>) = 48</p> <p>2 LSB (<math>16 \times 2</math>) = 32</p> <p>1 LSB (<math>16 \times 1</math>) = 16</p> <p>0 LSB (<math>16 \times 0</math>) = 0</p> <p>-1 LSB (<math>16 \times (-1)</math>) = 240</p> <p>:</p> <p>:</p> <p>-7LSB(<math>16 \times (-7)</math>) = 144</p> <p>Reset type: XRSn</p>

### 11.15.3.41 ADCPPB1CONFIG Register (Offset = 40h) [Reset = 0000h]

ADCPPB1CONFIG is shown in [Figure 11-81](#) and described in [Table 11-76](#).

Return to the [Summary Table](#).

ADC PPB1 Config Register

**Figure 11-81. ADCPPB1CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPEN	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 11-76. ADCPPB1CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 1 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB1RESULT register. 0 ADCPPB1RESULT = ADCRESULTx - ADCPPB1OFFREF 1 ADCPPB1RESULT = ADCPPB1OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 11-76. ADCPPB1CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 1 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.</p> <p>0000 SOC0/EOC0/RESULT0 is associated with post processing block 1</p> <p>0001 SOC1/EOC1/RESULT1 is associated with post processing block 1</p> <p>0010 SOC2/EOC2/RESULT2 is associated with post processing block 1</p> <p>0011 SOC3/EOC3/RESULT3 is associated with post processing block 1</p> <p>0100 SOC4/EOC4/RESULT4 is associated with post processing block 1</p> <p>0101 SOC5/EOC5/RESULT5 is associated with post processing block 1</p> <p>0110 SOC6/EOC6/RESULT6 is associated with post processing block 1</p> <p>0111 SOC7/EOC7/RESULT7 is associated with post processing block 1</p> <p>1000 SOC8/EOC8/RESULT8 is associated with post processing block 1</p> <p>1001 SOC9/EOC9/RESULT9 is associated with post processing block 1</p> <p>1010 SOC10/EOC10/RESULT10 is associated with post processing block 1</p> <p>1011 SOC11/EOC11/RESULT11 is associated with post processing block 1</p> <p>1100 SOC12/EOC12/RESULT12 is associated with post processing block 1</p> <p>1101 SOC13/EOC13/RESULT13 is associated with post processing block 1</p> <p>1110 SOC14/EOC14/RESULT14 is associated with post processing block 1</p> <p>1111 SOC15/EOC15/RESULT15 is associated with post processing block 1</p> <p>Reset type: SYSRSn</p>

### 11.15.3.42 ADCPPB1STAMP Register (Offset = 41h) [Reset = 0000h]

ADCPPB1STAMP is shown in [Figure 11-82](#) and described in [Table 11-77](#).

Return to the [Summary Table](#).

ADC PPB1 Sample Delay Time Stamp Register

**Figure 11-82. ADCPPB1STAMP Register**

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

**Table 11-77. ADCPPB1STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 1 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

### 11.15.3.43 ADCPPB1OFFCAL Register (Offset = 42h) [Reset = 0000h]

ADCPPB1OFFCAL is shown in [Figure 11-83](#) and described in [Table 11-78](#).

Return to the [Summary Table](#).

ADC PPB1 Offset Calibration Register

**Figure 11-83. ADCPPB1OFFCAL Register**

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

**Table 11-78. ADCPPB1OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 1 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.</p> <p>Reset type: SYSRSn</p>

### 11.15.3.44 ADCPPB1OFFREF Register (Offset = 43h) [Reset = 0000h]

ADCPPB1OFFREF is shown in [Figure 11-84](#) and described in [Table 11-79](#).

Return to the [Summary Table](#).

ADC PPB1 Offset Reference Register

**Figure 11-84. ADCPPB1OFFREF Register**

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

**Table 11-79. ADCPPB1OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	ADC Post Processing Block 1 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB1RESULT register. This subtraction is not saturated. 0000h No change. The ADCRESULT value is passed on. 0001h ADCRESULT - 1 is passed on. 0002h ADCRESULT - 2 is passed on. ... 8000h ADCRESULT - 32,768 is passed on. ... FFFFh ADCRESULT - 65,535 is passed on. NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode. Reset type: SYSRSn

### 11.15.3.45 ADCPPB1TRIPHI Register (Offset = 44h) [Reset = 0000000h]

ADCPPB1TRIPHI is shown in [Figure 11-85](#) and described in [Table 11-80](#).

Return to the [Summary Table](#).

ADC PPB1 Trip High Register

**Figure 11-85. ADCPPB1TRIPHI Register**

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

**Table 11-80. ADCPPB1TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 1 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB1RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB1RESULT register. Reset type: SYSRSn

### 11.15.3.46 ADCPPB1TRIPLO Register (Offset = 46h) [Reset = 0000000h]

ADCPPB1TRIPLO is shown in [Figure 11-86](#) and described in [Table 11-81](#).

Return to the [Summary Table](#).

ADC PPB1 Trip Low/Trigger Time Stamp Register

**Figure 11-86. ADCPPB1TRIPLO Register**

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

**Table 11-81. ADCPPB1TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 1 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 1 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB1RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRSULT bit field of the ADCPPB1RESULT register. Reset type: SYSRSn

### 11.15.3.47 ADCPPB2CONFIG Register (Offset = 48h) [Reset = 0000h]

ADCPPB2CONFIG is shown in [Figure 11-87](#) and described in [Table 11-82](#).

Return to the [Summary Table](#).

ADC PPB2 Config Register

**Figure 11-87. ADCPPB2CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 11-82. ADCPPB2CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 2 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB2RESULT register. 0 ADCPPB2RESULT = ADCRESULTx - ADCPPB2OFFREF 1 ADCPPB2RESULT = ADCPPB2OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 11-82. ADCPPB2CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	ADC Post Processing Block 2 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block. 0000 SOC0/EOC0/RESULT0 is associated with post processing block 2 0001 SOC1/EOC1/RESULT1 is associated with post processing block 2 0010 SOC2/EOC2/RESULT2 is associated with post processing block 2 0011 SOC3/EOC3/RESULT3 is associated with post processing block 2 0100 SOC4/EOC4/RESULT4 is associated with post processing block 2 0101 SOC5/EOC5/RESULT5 is associated with post processing block 2 0110 SOC6/EOC6/RESULT6 is associated with post processing block 2 0111 SOC7/EOC7/RESULT7 is associated with post processing block 2 1000 SOC8/EOC8/RESULT8 is associated with post processing block 2 1001 SOC9/EOC9/RESULT9 is associated with post processing block 2 1010 SOC10/EOC10/RESULT10 is associated with post processing block 2 1011 SOC11/EOC11/RESULT11 is associated with post processing block 2 1100 SOC12/EOC12/RESULT12 is associated with post processing block 2 1101 SOC13/EOC13/RESULT13 is associated with post processing block 2 1110 SOC14/EOC14/RESULT14 is associated with post processing block 2 1111 SOC15/EOC15/RESULT15 is associated with post processing block 2 Reset type: SYSRStn

### 11.15.3.48 ADCPPB2STAMP Register (Offset = 49h) [Reset = 0000h]

ADCPPB2STAMP is shown in [Figure 11-88](#) and described in [Table 11-83](#).

Return to the [Summary Table](#).

ADC PPB2 Sample Delay Time Stamp Register

**Figure 11-88. ADCPPB2STAMP Register**

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

**Table 11-83. ADCPPB2STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 2 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

### 11.15.3.49 ADCPPB2OFFCAL Register (Offset = 4Ah) [Reset = 0000h]

ADCPPB2OFFCAL is shown in [Figure 11-89](#) and described in [Table 11-84](#).

Return to the [Summary Table](#).

ADC PPB2 Offset Calibration Register

**Figure 11-89. ADCPPB2OFFCAL Register**

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

**Table 11-84. ADCPPB2OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	ADC Post Processing Block 2 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register. 000h No change. The ADC output is stored directly into ADCRESULT. 001h ADC output - 1 is stored into ADCRESULT. 002h ADC output - 2 is stored into ADCRESULT. ... 200h ADC output + 512 is stored into ADCRESULT. ... 3FFh ADC output + 1 is stored into ADCRESULT. NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register. Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied. Reset type: SYSRSn

### 11.15.3.50 ADCPPB2OFFREF Register (Offset = 4Bh) [Reset = 0000h]

ADCPPB2OFFREF is shown in [Figure 11-90](#) and described in [Table 11-85](#).

Return to the [Summary Table](#).

ADC PPB2 Offset Reference Register

**Figure 11-90. ADCPPB2OFFREF Register**

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

**Table 11-85. ADCPPB2OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 2 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB2RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.            0001h ADCRESULT - 1 is passed on.            0002h ADCRESULT - 2 is passed on.            ...            8000h ADCRESULT - 32,768 is passed on.            ...            FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.            Reset type: SYSRSn</p>

### 11.15.3.51 ADCPPB2TRIPHI Register (Offset = 4Ch) [Reset = 0000000h]

ADCPPB2TRIPHI is shown in [Figure 11-91](#) and described in [Table 11-86](#).

Return to the [Summary Table](#).

ADC PPB2 Trip High Register

**Figure 11-91. ADCPPB2TRIPHI Register**

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

**Table 11-86. ADCPPB2TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 2 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB2RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB2RESULT register. Reset type: SYSRSn

### 11.15.3.52 ADCPPB2TRIPLO Register (Offset = 4Eh) [Reset = 0000000h]

ADCPPB2TRIPLO is shown in [Figure 11-92](#) and described in [Table 11-87](#).

Return to the [Summary Table](#).

ADC PPB2 Trip Low/Trigger Time Stamp Register

**Figure 11-92. ADCPPB2TRIPLO Register**

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

**Table 11-87. ADCPPB2TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 2 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 2 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB2RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRSULT bit field of the ADCPPB2RESULT register. Reset type: SYSRSn

### 11.15.3.53 ADCPPB3CONFIG Register (Offset = 50h) [Reset = 0000h]

ADCPPB3CONFIG is shown in [Figure 11-93](#) and described in [Table 11-88](#).

Return to the [Summary Table](#).

ADC PPB3 Config Register

**Figure 11-93. ADCPPB3CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPE N	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 11-88. ADCPPB3CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 3 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB3RESULT register. 0 ADCPPB3RESULT = ADCRESULTx - ADCPPB3OFFREF 1 ADCPPB3RESULT = ADCPPB3OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 11-88. ADCPPB3CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	<p>ADC Post Processing Block 3 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block.</p> <p>0000 SOC0/EOC0/RESULT0 is associated with post processing block 3</p> <p>0001 SOC1/EOC1/RESULT1 is associated with post processing block 3</p> <p>0010 SOC2/EOC2/RESULT2 is associated with post processing block 3</p> <p>0011 SOC3/EOC3/RESULT3 is associated with post processing block 3</p> <p>0100 SOC4/EOC4/RESULT4 is associated with post processing block 3</p> <p>0101 SOC5/EOC5/RESULT5 is associated with post processing block 3</p> <p>0110 SOC6/EOC6/RESULT6 is associated with post processing block 3</p> <p>0111 SOC7/EOC7/RESULT7 is associated with post processing block 3</p> <p>1000 SOC8/EOC8/RESULT8 is associated with post processing block 3</p> <p>1001 SOC9/EOC9/RESULT9 is associated with post processing block 3</p> <p>1010 SOC10/EOC10/RESULT10 is associated with post processing block 3</p> <p>1011 SOC11/EOC11/RESULT11 is associated with post processing block 3</p> <p>1100 SOC12/EOC12/RESULT12 is associated with post processing block 3</p> <p>1101 SOC13/EOC13/RESULT13 is associated with post processing block 3</p> <p>1110 SOC14/EOC14/RESULT14 is associated with post processing block 3</p> <p>1111 SOC15/EOC15/RESULT15 is associated with post processing block 3</p> <p>Reset type: SYSRSn</p>

### 11.15.3.54 ADCPPB3STAMP Register (Offset = 51h) [Reset = 0000h]

ADCPPB3STAMP is shown in [Figure 11-94](#) and described in [Table 11-89](#).

Return to the [Summary Table](#).

ADC PPB3 Sample Delay Time Stamp Register

**Figure 11-94. ADCPPB3STAMP Register**

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

**Table 11-89. ADCPPB3STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 3 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

### 11.15.3.55 ADCPPB3OFFCAL Register (Offset = 52h) [Reset = 0000h]

ADCPPB3OFFCAL is shown in [Figure 11-95](#) and described in [Table 11-90](#).

Return to the [Summary Table](#).

ADC PPB3 Offset Calibration Register

**Figure 11-95. ADCPPB3OFFCAL Register**

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

**Table 11-90. ADCPPB3OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	<p>ADC Post Processing Block 3 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register.</p> <p>000h No change. The ADC output is stored directly into ADCRESULT.</p> <p>001h ADC output - 1 is stored into ADCRESULT.</p> <p>002h ADC output - 2 is stored into ADCRESULT.</p> <p>...</p> <p>200h ADC output + 512 is stored into ADCRESULT.</p> <p>...</p> <p>3FFh ADC output + 1 is stored into ADCRESULT.</p> <p>NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register.</p> <p>Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied.</p> <p>Reset type: SYSRSn</p>

### 11.15.3.56 ADCPPB3OFFREF Register (Offset = 53h) [Reset = 0000h]

ADCPPB3OFFREF is shown in [Figure 11-96](#) and described in [Table 11-91](#).

Return to the [Summary Table](#).

ADC PPB3 Offset Reference Register

**Figure 11-96. ADCPPB3OFFREF Register**

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

**Table 11-91. ADCPPB3OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	ADC Post Processing Block 3 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB3RESULT register. This subtraction is not saturated. 0000h No change. The ADCRESULT value is passed on. 0001h ADCRESULT - 1 is passed on. 0002h ADCRESULT - 2 is passed on. ... 8000h ADCRESULT - 32,768 is passed on. ... FFFFh ADCRESULT - 65,535 is passed on. NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode. Reset type: SYSRSn

### 11.15.3.57 ADCPPB3TRIPHI Register (Offset = 54h) [Reset = 0000000h]

ADCPPB3TRIPHI is shown in [Figure 11-97](#) and described in [Table 11-92](#).

Return to the [Summary Table](#).

ADC PPB3 Trip High Register

**Figure 11-97. ADCPPB3TRIPHI Register**

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

**Table 11-92. ADCPPB3TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 3 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB3RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB3RESULT register. Reset type: SYSRSn

### 11.15.3.58 ADCPPB3TRIPLO Register (Offset = 56h) [Reset = 0000000h]

ADCPPB3TRIPLO is shown in [Figure 11-98](#) and described in [Table 11-93](#).

Return to the [Summary Table](#).

ADC PPB3 Trip Low/Trigger Time Stamp Register

**Figure 11-98. ADCPPB3TRIPLO Register**

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

**Table 11-93. ADCPPB3TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 3 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 3 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB3RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB3RESULT register. Reset type: SYSRSn

### 11.15.3.59 ADCPPB4CONFIG Register (Offset = 58h) [Reset = 0000h]

ADCPPB4CONFIG is shown in [Figure 11-99](#) and described in [Table 11-94](#).

Return to the [Summary Table](#).

ADC PPB4 Config Register

**Figure 11-99. ADCPPB4CONFIG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CBCEN	TWOSCOMPEN	CONFIG			
R-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 11-94. ADCPPB4CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	Reserved
5	CBCEN	R/W	0h	ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present. Reset type: SYSRSn
4	TWOSCOMPEN	R/W	0h	ADC Post Processing Block 4 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB4RESULT register. 0 ADCPPB4RESULT = ADCRESULTx - ADCPPB4OFFREF 1 ADCPPB4RESULT = ADCPPB4OFFREF - ADCRESULTx Reset type: SYSRSn

**Table 11-94. ADCPPB4CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CONFIG	R/W	0h	ADC Post Processing Block 4 Configuration. This bit field defines which SOC/EOC/RESULT is associated with this post processing block. 0000 SOC0/EOC0/RESULT0 is associated with post processing block 4 0001 SOC1/EOC1/RESULT1 is associated with post processing block 4 0010 SOC2/EOC2/RESULT2 is associated with post processing block 4 0011 SOC3/EOC3/RESULT3 is associated with post processing block 4 0100 SOC4/EOC4/RESULT4 is associated with post processing block 4 0101 SOC5/EOC5/RESULT5 is associated with post processing block 4 0110 SOC6/EOC6/RESULT6 is associated with post processing block 4 0111 SOC7/EOC7/RESULT7 is associated with post processing block 4 1000 SOC8/EOC8/RESULT8 is associated with post processing block 4 1001 SOC9/EOC9/RESULT9 is associated with post processing block 4 1010 SOC10/EOC10/RESULT10 is associated with post processing block 4 1011 SOC11/EOC11/RESULT11 is associated with post processing block 4 1100 SOC12/EOC12/RESULT12 is associated with post processing block 4 1101 SOC13/EOC13/RESULT13 is associated with post processing block 4 1110 SOC14/EOC14/RESULT14 is associated with post processing block 4 1111 SOC15/EOC15/RESULT15 is associated with post processing block 4 Reset type: SYSRStn

### 11.15.3.60 ADCPPB4STAMP Register (Offset = 59h) [Reset = 0000h]

ADCPPB4STAMP is shown in [Figure 11-100](#) and described in [Table 11-95](#).

Return to the [Summary Table](#).

ADC PPB4 Sample Delay Time Stamp Register

**Figure 11-100. ADCPPB4STAMP Register**

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

**Table 11-95. ADCPPB4STAMP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DLYSTAMP	R	0h	ADC Post Processing Block 4 Delay Time Stamp. When an SOC starts sampling the value contained in REQSTAMP is subtracted from the value in ADCCOUNTER.FREECOUNT and loaded into this bit field, thereby giving the number of system clock cycles delay between the SOC trigger and the actual start of the sample. Reset type: SYSRSn

### 11.15.3.61 ADCPPB4OFFCAL Register (Offset = 5Ah) [Reset = 0000h]

ADCPPB4OFFCAL is shown in [Figure 11-101](#) and described in [Table 11-96](#).

Return to the [Summary Table](#).

ADC PPB4 Offset Calibration Register

**Figure 11-101. ADCPPB4OFFCAL Register**

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

**Table 11-96. ADCPPB4OFFCAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	OFFCAL	R/W	0h	ADC Post Processing Block 4 Offset Correction. This bit field can be used to digitally remove any system level offset inherent in the ADCIN circuit. This 10-bit signed value is subtracted from the ADC output before being stored in the ADCRESULT register. 000h No change. The ADC output is stored directly into ADCRESULT. 001h ADC output - 1 is stored into ADCRESULT. 002h ADC output - 2 is stored into ADCRESULT. ... 200h ADC output + 512 is stored into ADCRESULT. ... 3FFh ADC output + 1 is stored into ADCRESULT. NOTE: In 16-bit mode, the subtraction will saturate at 0000h and FFFFh before being stored into the ADCRESULT register. In 12-bit mode, the subtraction will saturate at 0000h and 0FFFh before being stored into the ADCRESULT register. Note: in the case that multiple PPBs point to the same SOC, only the OFFCAL of the highest numbered PPB will be applied. Reset type: SYSRSn

### 11.15.3.62 ADCPPB4OFFREF Register (Offset = 5Bh) [Reset = 0000h]

ADCPPB4OFFREF is shown in [Figure 11-102](#) and described in [Table 11-97](#).

Return to the [Summary Table](#).

ADC PPB4 Offset Reference Register

**Figure 11-102. ADCPPB4OFFREF Register**

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

**Table 11-97. ADCPPB4OFFREF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	OFFREF	R/W	0h	<p>ADC Post Processing Block 4 Offset Correction. This bit field can be used to either calculate the feedback error or convert a unipolar signal to bipolar by subtracting a reference value. This 16-bit unsigned value is subtracted from the ADCRESULT register before being passed through an optional two's complement function and stored in the ADCPPB4RESULT register. This subtraction is not saturated.</p> <p>0000h No change. The ADCRESULT value is passed on.            0001h ADCRESULT - 1 is passed on.            0002h ADCRESULT - 2 is passed on.            ...            8000h ADCRESULT - 32,768 is passed on.            ...            FFFFh ADCRESULT - 65,535 is passed on.</p> <p>NOTE: In 12-bit mode the size of this register does not change from 16-bits. It is the user's responsibility to ensure that only a 12-bit value is written to this register when in 12-bit mode.            Reset type: SYSRSn</p>

### 11.15.3.63 ADCPPB4TRIPHI Register (Offset = 5Ch) [Reset = 0000000h]

ADCPPB4TRIPHI is shown in [Figure 11-103](#) and described in [Table 11-98](#).

Return to the [Summary Table](#).

ADC PPB4 Trip High Register

**Figure 11-103. ADCPPB4TRIPHI Register**

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

**Table 11-98. ADCPPB4TRIPHI Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	HSIGN	R/W	0h	High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITHI	R/W	0h	ADC Post Processing Block 4 Trip High Limit. This value sets the digital comparator trip high limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB4RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB4RESULT register. Reset type: SYSRSn

### 11.15.3.64 ADCPPB4TRIPLO Register (Offset = 5Eh) [Reset = 0000000h]

ADCPPB4TRIPLO is shown in [Figure 11-104](#) and described in [Table 11-99](#).

Return to the [Summary Table](#).

ADC PPB4 Trip Low/Trigger Time Stamp Register

**Figure 11-104. ADCPPB4TRIPLO Register**

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

**Table 11-99. ADCPPB4TRIPLO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	REQSTAMP	R	0h	ADC Post Processing Block 4 Request Time Stamp. When a trigger sets the associated SOC flag in the ADCSOCFLG1 register the value of ADCCOUNTER.FREECOUNT is loaded into this bit field. Reset type: SYSRSn
19-17	RESERVED	R	0h	Reserved
16	LSIGN	R/W	0h	Low Limit Sign Bit. This is the sign bit (17th bit) to the LIMITLO bit field when in 16-bit ADC mode. Reset type: SYSRSn
15-0	LIMITLO	R/W	0h	ADC Post Processing Block 4 Trip Low Limit. This value sets the digital comparator trip low limit. In 16-bit mode all 17 bits will be compared against the 17 bits of the PPBRESULT bit field of the ADCPPB4RESULT register. In 12-bit mode bits 12:0 will be compared against bits 12:0 of the PPBRESULT bit field of the ADCPPB4RESULT register. Reset type: SYSRSn

### 11.15.3.65 ADCINTCYCLE Register (Offset = 6Fh) [Reset = 0000h]

ADCINTCYCLE is shown in [Figure 11-105](#) and described in [Table 11-100](#).

Return to the [Summary Table](#).

ADC Early Interrupt Generation Cycle

**Figure 11-105. ADCINTCYCLE Register**

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

**Table 11-100. ADCINTCYCLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DELAY	R/W	0h	ADC Early Interrupt Generation Cycle Delay: Defines the delay from the fall edge of ADCSOC in terms of system clock cycles, for the interrupt to be generated. Reset type: SYSRSn

### 11.15.3.66 ADCINLTRIM2 Register (Offset = 72h) [Reset = 0000000h]

ADCINLTRIM2 is shown in [Figure 11-106](#) and described in [Table 11-101](#).

Return to the [Summary Table](#).

ADC Linearity Trim 2 Register

**Figure 11-106. 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																															

**Table 11-101. 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. Reset type: XRSn

### 11.15.3.67 ADCINLTRIM3 Register (Offset = 74h) [Reset = 0000000h]

ADCINLTRIM3 is shown in [Figure 11-107](#) and described in [Table 11-102](#).

Return to the [Summary Table](#).

ADC Linearity Trim 3 Register

**Figure 11-107. 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																															

**Table 11-102. 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. Reset type: XRSn

### 11.15.4 ADC Registers to Driverlib Functions

**Table 11-103. ADC Registers to Driverlib Functions**

File	Driverlib Function
<b>ADCCTL1</b>	
adc.h	ADC_setInterruptPulseMode
adc.h	ADC_enableConverter
adc.h	ADC_disableConverter
adc.h	ADC_isBusy
<b>ADCCTL2</b>	
adc.h	ADC_setPrescaler
<b>ADCBURSTCTL</b>	
adc.h	ADC_setBurstModeConfig
adc.h	ADC_enableBurstMode
adc.h	ADC_disableBurstMode
<b>ADCINTFLG</b>	
adc.h	ADC_getInterruptStatus
adc.h	ADC_clearInterruptStatus
<b>ADCINTFLGCLR</b>	
adc.h	ADC_clearInterruptStatus
<b>ADCINTOVF</b>	
adc.h	ADC_getInterruptOverflowStatus
adc.h	ADC_clearInterruptOverflowStatus
<b>ADCINTOVFCLR</b>	
adc.h	ADC_clearInterruptOverflowStatus
<b>ADCINTSEL1N2</b>	
adc.h	ADC_enableInterrupt
adc.h	ADC_disableInterrupt
adc.h	ADC_setInterruptSource
adc.h	ADC_enableContinuousMode

**Table 11-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
adc.h	ADC_disableContinuousMode
<b>ADCINTSEL3N4</b>	
-	See INTSEL1N2
<b>ADCSOCPRCTL</b>	
adc.h	ADC_setSOCPriority
<b>ADCINTSOCSEL1</b>	
adc.h	ADC_setInterruptSOCTrigger
<b>ADCINTSOCSEL2</b>	
-	See INTSOCSEL1
<b>ADCSOCFLG1</b>	
-	
<b>ADCSOCFRC1</b>	
adc.h	ADC_forceSOC
adc.h	ADC_forceMultipleSOC
<b>ADCSOCOVF1</b>	
-	
<b>ADCSOCOVFCLR1</b>	
-	
<b>ADCSOC0CTL</b>	
adc.h	ADC_setupSOC
<b>ADCSOC1CTL</b>	
-	See SOC0CTL
<b>ADCSOC2CTL</b>	
-	See SOC0CTL
<b>ADCSOC3CTL</b>	
-	See SOC0CTL
<b>ADCSOC4CTL</b>	
-	See SOC0CTL
<b>ADCSOC5CTL</b>	
-	See SOC0CTL
<b>ADCSOC6CTL</b>	
-	See SOC0CTL
<b>ADCSOC7CTL</b>	
-	See SOC0CTL
<b>ADCSOC8CTL</b>	
-	See SOC0CTL
<b>ADCSOC9CTL</b>	
-	See SOC0CTL
<b>ADCSOC10CTL</b>	
-	See SOC0CTL
<b>ADCSOC11CTL</b>	
-	See SOC0CTL
<b>ADCSOC12CTL</b>	
-	See SOC0CTL
<b>ADCSOC13CTL</b>	

**Table 11-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See SOC0CTL
<b>ADCSOC14CTL</b>	
-	See SOC0CTL
<b>ADCSOC15CTL</b>	
-	See SOC0CTL
<b>ADCEVTSTAT</b>	
adc.h	ADC_getPPBEventStatus
<b>ADCEVTCLR</b>	
adc.h	ADC_clearPPBEventStatus
<b>ADCEVTSEL</b>	
adc.h	ADC_enablePPBEvent
adc.h	ADC_disablePPBEvent
<b>ADCEVTINTSEL</b>	
adc.h	ADC_enablePPBEventInterrupt
adc.h	ADC_disablePPBEventInterrupt
<b>ADCOSDETECT</b>	
adc.h	ADC_configOSDetectMode
<b>ADCCOUNTER</b>	
-	
<b>ADCREV</b>	
-	
<b>ADCOFFTRIM</b>	
adc.c	ADC_setOffsetTrim
adc.c	ADC_setOffsetTrimAll
<b>ADCPPB1CONFIG</b>	
adc.h	ADC_setupPPB
adc.h	ADC_enablePPBEventCBCClear
adc.h	ADC_disablePPBEventCBCClear
adc.h	ADC_enablePPBTwosComplement
adc.h	ADC_disablePPBTwosComplement
<b>ADCPPB1STAMP</b>	
adc.h	ADC_getPPBDelayTimeStamp
<b>ADCPPB1OFFCAL</b>	
adc.h	ADC_setPPBCalibrationOffset
<b>ADCPPB1OFFREF</b>	
adc.h	ADC_setPPBReferenceOffset
<b>ADCPPB1TRIPHI</b>	
adc.c	ADC_setPPBTripLimits
<b>ADCPPB1TRIPLO</b>	
adc.c	ADC_setPPBTripLimits
<b>ADCPPB2CONFIG</b>	
-	See PPB1CONFIG
<b>ADCPPB2STAMP</b>	
-	See PPB1STAMP
<b>ADCPPB2OFFCAL</b>	

**Table 11-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	See PPB1OFFCAL
<b>ADCPPB2OFFREF</b>	
-	See PPB1OFFREF
<b>ADCPPB2TRIPHI</b>	
-	See PPB1TRIPHI
<b>ADCPPB2TRIPLO</b>	
-	See PPB1TRIPLO
<b>ADCPPB3CONFIG</b>	
-	See PPB1CONFIG
<b>ADCPPB3STAMP</b>	
-	See PPB1STAMP
<b>ADCPPB3OFFCAL</b>	
-	See PPB1OFFCAL
<b>ADCPPB3OFFREF</b>	
-	See PPB1OFFREF
<b>ADCPPB3TRIPHI</b>	
-	See PPB1TRIPHI
<b>ADCPPB3TRIPLO</b>	
-	See PPB1TRIPLO
<b>ADCPPB4CONFIG</b>	
-	See PPB1CONFIG
<b>ADCPPB4STAMP</b>	
-	See PPB1STAMP
<b>ADCPPB4OFFCAL</b>	
-	See PPB1OFFCAL
<b>ADCPPB4OFFREF</b>	
-	See PPB1OFFREF
<b>ADCPPB4TRIPHI</b>	
-	See PPB1TRIPHI
<b>ADCPPB4TRIPLO</b>	
-	See PPB1TRIPLO
<b>ADCINTCYCLE</b>	
adc.h	ADC_setInterruptCycleOffset
<b>ADCINLTRIM2</b>	
adc.c	ADC_setINLTrim
<b>ADCINLTRIM3</b>	
-	
<b>ADCRESULT0</b>	
adc.h	ADC_readResult
<b>ADCRESULT1</b>	
-	See RESULT0
<b>ADCRESULT2</b>	
-	See RESULT0
<b>ADCRESULT3</b>	
-	See RESULT0

**Table 11-103. ADC Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>ADCRESULT4</b>	
-	See RESULT0
<b>ADCRESULT5</b>	
-	See RESULT0
<b>ADCRESULT6</b>	
-	See RESULT0
<b>ADCRESULT7</b>	
-	See RESULT0
<b>ADCRESULT8</b>	
-	See RESULT0
<b>ADCRESULT9</b>	
-	See RESULT0
<b>ADCRESULT10</b>	
-	See RESULT0
<b>ADCRESULT11</b>	
-	See RESULT0
<b>ADCRESULT12</b>	
-	See RESULT0
<b>ADCRESULT13</b>	
-	See RESULT0
<b>ADCRESULT14</b>	
-	See RESULT0
<b>ADCRESULT15</b>	
-	See RESULT0
<b>ADCPPB1RESULT</b>	
adc.h	ADC_readPPBResult
<b>ADCPPB2RESULT</b>	
-	See PPB1RESULT
<b>ADCPPB3RESULT</b>	
-	See PPB1RESULT
<b>ADCPPB4RESULT</b>	
-	See PPB1RESULT

## Chapter 12 Comparator Subsystem (CMPSS)



The Comparator Subsystem (CMPSS) consists of analog comparators and supporting circuits that are useful for power applications such as peak current mode control, switched-mode power, power factor correction, voltage trip monitoring, and so forth.

This device contains two variants of the CMPSS module: CMPSS and CMPSS\_LITE. These modules share a common architecture, but some features are supported only by the full CMPSS variant and not the CMPSS\_LITE variant. Carefully note where the documentation indicates that these certain features are not supported by the CMPSS\_LITE variant. Also note that the device data sheet indicates electrical characteristics for the two variants separately.

See the [C2000 Real-Time Control Peripheral Reference Guide](#) 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.

<b>12.1 Introduction</b> .....	<b>1186</b>
<b>12.2 Comparator</b> .....	<b>1188</b>
<b>12.3 Reference DAC</b> .....	<b>1189</b>
<b>12.4 Ramp Generator</b> .....	<b>1190</b>
<b>12.5 Digital Filter</b> .....	<b>1193</b>
<b>12.6 Using the CMPSS</b> .....	<b>1194</b>
<b>12.7 CMPSS DAC Output</b> .....	<b>1196</b>
<b>12.8 Software</b> .....	<b>1197</b>
<b>12.9 CMPSS Registers</b> .....	<b>1198</b>

## 12.1 Introduction

The comparator subsystem is built around a number of modules. Each subsystem contains two comparators, two reference 12-bit DACs, and two digital filters. The subsystem also includes one ramp generator (full CMPSS modules only; not supported by CMPSS\_LITE instances). The ramp generator ramps down only. Comparators are denoted "H" or "L" within each module where "H" and "L" represent high and low, respectively. Each comparator generates a digital output which indicates whether the voltage on the positive input is greater than the voltage on the negative input. The positive input of the comparator is driven from an external pin (see the *Analog Subsystem* chapter for mux options available to the CMPSS). The negative input can be driven by an external pin or by the programmable reference 12-bit DAC. Each comparator output passes through a programmable digital filter that can remove spurious trip signals. An unfiltered output is also available if filtering is not required.

A ramp generator circuit is optionally available to control the reference 12-bit DAC value for the high comparator in the subsystem (full CMPSS modules only; not supported by CMPSS\_LITE instances).

### 12.1.1 CMPSS Related Collateral

#### Foundational Materials

- [C2000 Academy - CMPSS](#)
- [Real-Time Control Reference Guide](#)
  - Refer to the Comparator section

#### Expert Materials

- [Peak Current Control Realization for Boost Circuit Based On C2000™ MCU Application Report](#)
- [Peak Current Mode Controlled PSFB Converter Reference Design Using C2000™ Real-time MCU](#)
- [Understanding and Applying Current-Mode Control Theory Application Report](#)

### 12.1.2 Features

Each CMPSS includes:

- Two analog comparators
- Two programmable reference 12-bit DACs (full CMPSS only; CMPSS\_LITE instances are 12-bit with a lower effective resolution as described in the data manual)
- One decrementing ramp generator (full CMPSS only; not available on CMPSS\_LITE instances)
- Two digital filters, max filter clock prescale =  $2^{16}$
- Ability to synchronize submodules with EPWMSYNCPER
- Ability to extend clear signal with EPWMBLANK
- Ability to synchronize output with SYSCLK
- Ability to latch output
- Ability to invert output
- Option to use hysteresis on the input
- Option for negative input of comparator to be driven by an external signal or by the reference DAC
- VDDA is the DAC reference voltage
- Option to use the low comparator DAC output on an external pin (select instances only, mutually exclusive with use of compare functionality)

### 12.1.3 CMPSS Module Variants

This device contains two different variants of the CMPSS module: CMPSS (full module) and the CMPSS\_LITE (reduced functionality and performance). The differences in features between the two variants are summarized in Table 12-1.

Table 12-1. CMPSS and CMPSS\_LITE Feature Comparison

Feature	CMPSS	CMPSS_LITE
High and low comparators	Yes	Yes
Dual 12-bit reference DACs	Yes	Yes
DAC ramp generation	Yes	No
Low DAC output on external pin	Yes (Some instances)	No
Digital filters	Yes	Yes
Performance	Full performance (See CMPSS electrical characteristics in data sheet)	Some reduced performance (See CMPSS_LITE electrical characteristics in data sheet)

### 12.1.4 Block Diagram

The block diagram for the CMPSS is shown in Figure 12-1.

- CTRIPx(x= "H" or "L") signals are connected to the ePWM X-BAR for ePWM trip response. See the *Enhanced Pulse Width Modulator (ePWM)* chapter for more details on the ePWM X-BAR mux configuration.
- CTRIPxOUTx(x= "H" or "L") signals are connected to the Output X-BAR for external signaling. See the *General-Purpose Input/Output (GPIO)* chapter for more details on the Output X-BAR mux configuration.
- CMPxDACL is the low comparator DAC output which is available only in CMPSS1 module. To enable this DAC output, set the register CMPxDACOUTEN from analog subsystem.

#### Note

Enabling the DACL to a pin disables all other functionality: DACH, both COMP, the Ramp Generator, and the digital filters.

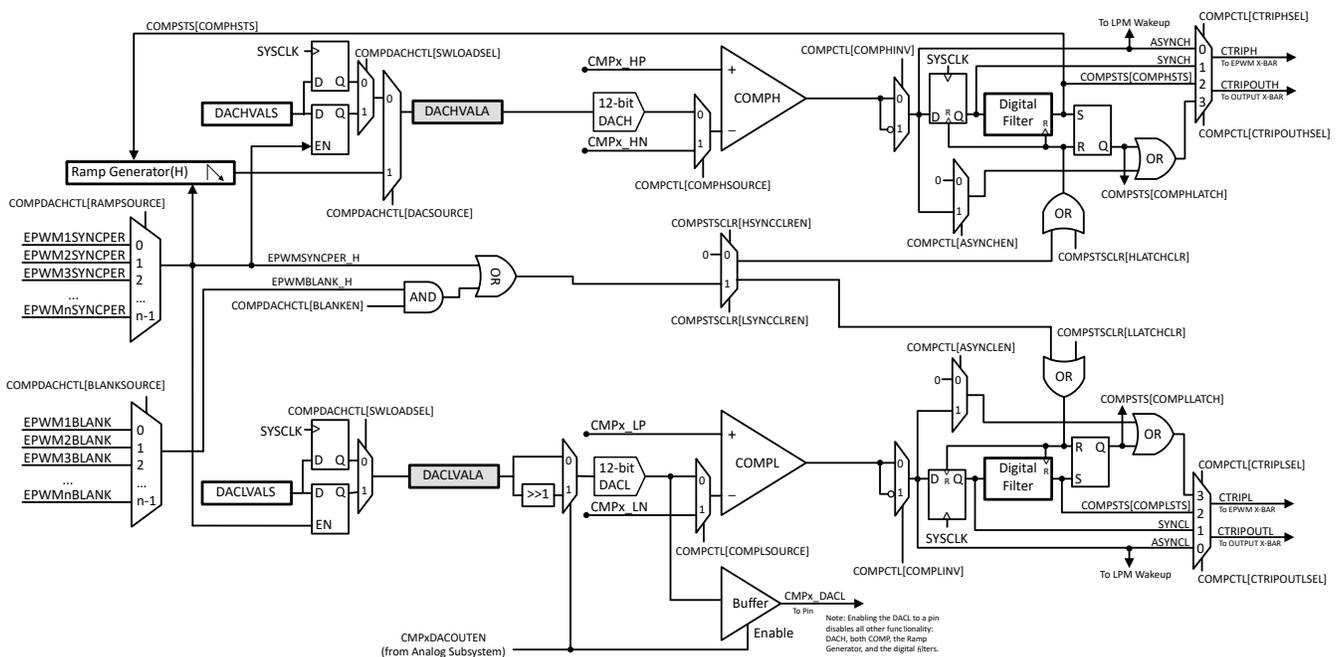


Figure 12-1. CMPSS Module Block Diagram



### 12.3 Reference DAC

Each reference 12-bit DAC can be configured to drive a reference voltage into the negative input of the respective comparator. Some CMPSS instances also allow the low DAC output to be routed to a pin to act as an external DAC. In this case, all other CMPSS module functionality is not useable, including the high DAC, both comparators, ramp generation, and the digital filters.

Two sets of DACxVAL registers, DACxVALA and DACxVALS, are present for each reference 12-bit DAC. DACxVALA is a read-only register that actively controls the reference 12-bit DAC value. DACxVALS is a writable shadow register that loads into DACxVALA either immediately or synchronized with the next EPWMSYNCPER event. The high and low reference 12-bit DAC (DACx) can optionally source the register DACxVALA value from the ramp generator instead of the register DACxVALS.

The operating range of the reference 12-bit DAC is bounded by DACREF and VSSA.

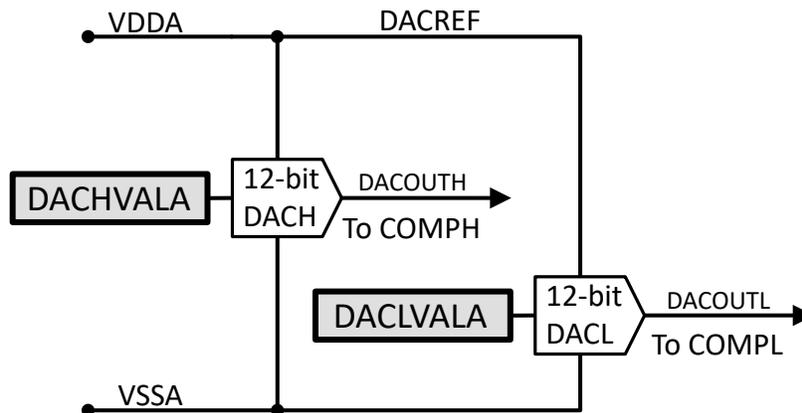


Figure 12-4. Reference DAC Block Diagram

The output of the reference 12-bit DAC can be calculated as:

$$DACOUT = \frac{(1 + DACVALA) * DACREF}{4096} \quad (10)$$

#### Note

- In the situations where both the DACH and DACL are driving the high and low comparators, a trip on one comparator can temporarily disturb the DAC output of the other comparator. The amount and length of time of this disturbance is specified in the device data sheet as “CMPSS DAC output disturbance” and “CMPSS DAC disturbance time”, respectively.

Users must design the system carefully so that if the input signal crosses either DACH or DACL and trips the associated comparator, the input signal stays more than a “CMPSS DAC output disturbance” away from the other comparator trip point for “CMPSS DAC disturbance time”.

- The DACH setting must always be higher than the DACL setting. If the user is not using the DACL, the DACLVALS register must be set to 0 to avoid the comparator COMPL from tripping and affecting the DACH. In this case, there is no limitation on the DACHVALS setting. Accordingly, when not using the DACH, the user must set the DACHVALS register to the maximum.
- The CMPSS instance can be enabled before programming the reference DAC values.

## 12.4 Ramp Generator

This section discusses the characteristics and behavior of the ramp generator.

### 12.4.1 Ramp Generator Overview

The ramp generator produces a falling ramp input for the high-reference 12-bit DAC when selected. In this mode, the reference 12-bit DAC uses the most-significant 12 bits of the RAMPSTS countdown register as the input. The low 4 bits of the RAMPSTS countdown register effectively act as a prescale for the falling ramp rate configurable with RAMPxSTEPVALA.

The ramp generator is enabled by setting DACSOURCE = 1. When DACSOURCE = 1 is selected, the value of RAMPSTS is loaded from RAMPxREFS and the register remains static until the selected EPWMSYNCPER signal is received. After receiving the selected EPWMSYNCPER signal, the value of RAMPxSTEPVALA is subtracted from RAMPSTS on every subsequent SYSCLK cycle.

To prevent the subtraction from commencing a SYSCLK cycle after a EPWMSYNCPER event, the RAMPDLYA register that serves as a delay counter can be used to hold off the RAMPSTS subtraction. On receiving a EPWMSYNCPER event, the value of RAMPDLYA is decremented by one on every SYSCLK cycle until the register reaches zero. So, the RAMPSTS subtraction only begins when RAMPDLYA is zero.

---

#### Note

- The ramp generation logic is only available on full CMPSS module instances; this function is not supported by CMPSS\_LITE instances.
-



### 12.4.3 Ramp Generator Behavior at Corner Cases

Since the ramp generator makes state changes on every rising edge of EPWMSYNCPER\_H and COMPHSTS, the following behavior can be expected on instances when these two events occur simultaneously or very close together.

Case 1: COMPHSTS rising edge occurs one or more cycles before EPWMSYNCPER\_H rising edge. RAMPSTS stops decrementing on COMPHSTS rising edge event. RAMPSTS starts decrementing on EPWMSYNCPER\_H rising edge event when RAMPDLYA reaches 0.

Case 2: COMPHSTS rising edge occurs simultaneously as EPWMSYNCPER\_H rising edge. EPWMSYNCPER rising edge event takes precedence and RAMPSTS starts decrementing when RAMPDLYA reaches 0. COMPHSTS rising edge event is ignored and does not halt RAMPSTS.

Case 3: COMPHSTS rising edge occurs one or more cycles after EPWMSYNCPER\_H rising edge but before RAMPDLYA reaches 0. RAMPSTS does not decrement when RAMPDLYA reaches 0.

Case 4: COMPHSTS rising edge occurs simultaneously as RAMPDLYA reaches 0 from EPWMSYNCPER\_H rising edge. RAMPSTS does not decrement.

This behavior is also illustrated in [Figure 12-6](#).

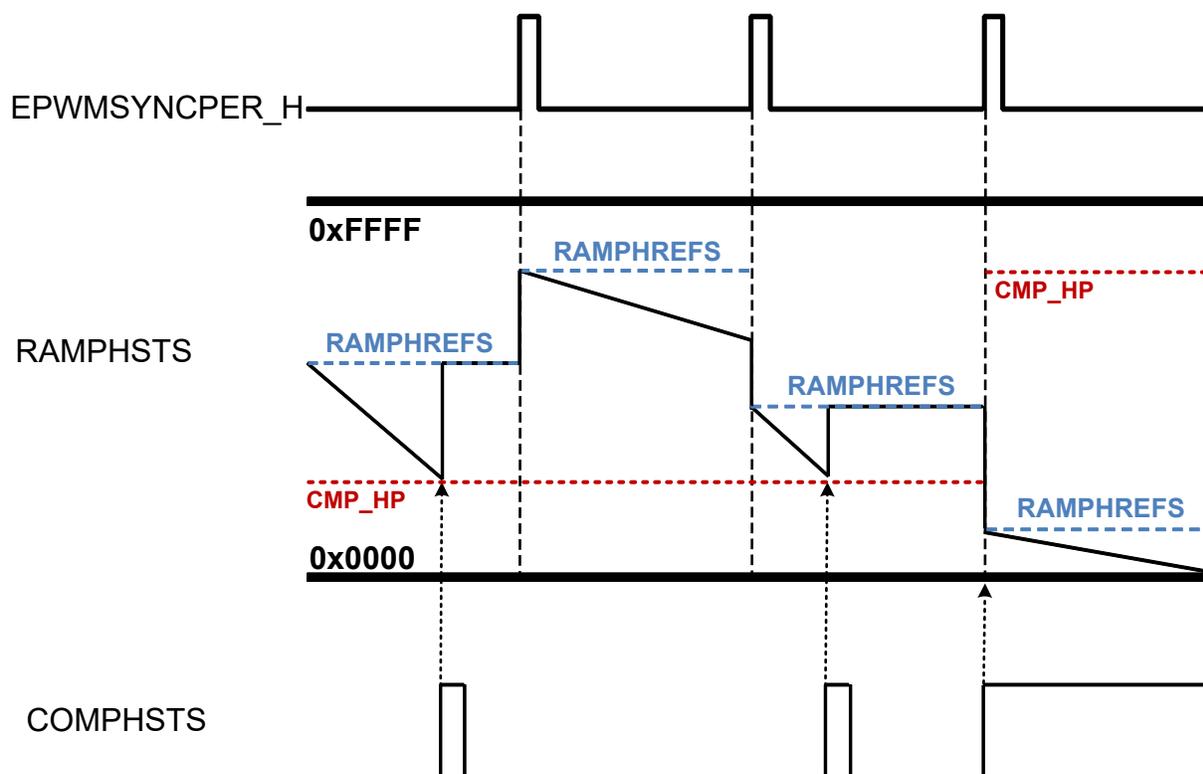


Figure 12-6. Ramp Generator Behavior

## 12.5 Digital Filter

The digital filter works on a window of FIFO samples (SAMPWIN) taken from the input. The filter output resolves to the majority value of the sample window, where majority is defined by the threshold (THRESH) value. If the majority threshold is not satisfied, the filter output remains unchanged.

For proper operation, the value of THRESH must be greater than  $SAMPWIN / 2$  and less than or equal to SAMPWIN.

A prescale function (CLKPRESCALE) determines the filter sampling rate, where the filter FIFO captures one sample every prescale system clocks. Old data from the FIFO is discarded.

Note that for SAMPWIN, THRESH and CLKPRESCALE, the internal number used by the digital filter is + 1 in all cases. In essence, samples = SAMPWIN + 1, threshold = THRESH + 1 and prescale = CLKPRESCALE + 1.

A conceptual model of the digital filter is shown in Figure 12-7.

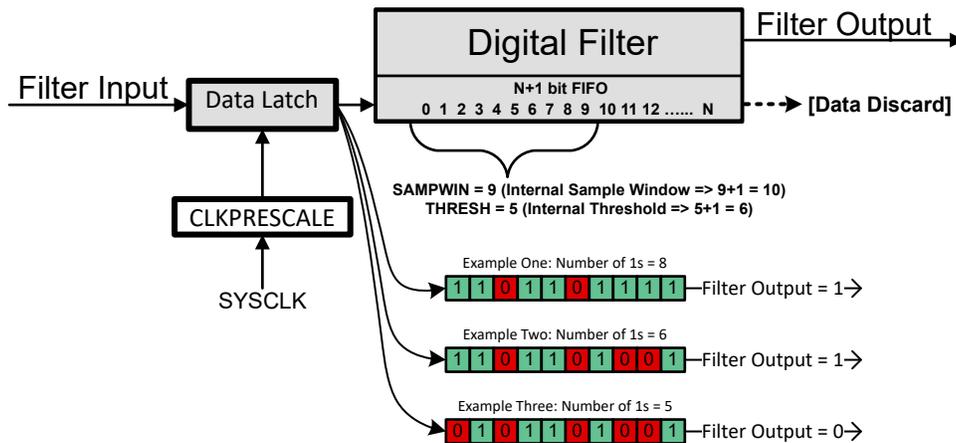


Figure 12-7. Digital Filter Behavior

Equivalent C code of the filter implementation is:

```

if (FILTER_OUTPUT == 0) {
    if (Num_1s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 1;
    }
}
else {
    if (Num_0s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 0;
    }
}

```

### 12.5.1 Filter Initialization Sequence

For proper operation of the digital filter, the following initialization sequence is recommended:

1. Configure and enable the comparator for operation.
2. Configure the digital filter parameters for operation:
  - Set SAMPWIN for the number of samples to monitor in the FIFO window.
  - Set THRESH for the threshold required for majority qualification.
  - Set CLKPRESCALE for the digital filter clock prescale value.
3. Initialize the sample values in the digital FIFO window by setting FILINIT.
4. Clear COMPSTS latch using COMPSTSCLR, if the latched path is desired.
5. Configure the CTRIP and CTRIPOUT signal paths.
6. If desired, configure the destination module, for example, ePWM, GPIO, and so on to accept the filtered signals.

## 12.6 Using the CMPSS

### 12.6.1 LATCHCLR, EPWMSYNCPER, and EPWMBLANK Signals

The LATCHCLR signal holds the digital filter, synchronization block, and the latch output in reset (0) after the required delays. The LATCHCLR signal is activated in software using xLATCHCLR (x = H or L). The LATCHCLR signal can also be activated by EPWMSYNCPER when xSYNCCLREN (x = H or L) is set. If a longer LATCHCLR signal is required, the EPWMBLANK signal can be used to extend the LATCHCLR signal by setting BLANKEN.

EPWMSYNCPER and EPWMBLANK (BLANKWDW) come from the Time-Base and Digital Compare submodules of the EPWM, respectively. For a detailed description of how these two signals are generated, refer to the respective submodule section in the *Enhanced Pulse Width Modulator (ePWM)* chapter.

The EPWMSYNCPER signal that loads DACxVALA when COMPDACCTL[SWLOADSEL] = 1 is a level trigger load. If TBCTR and TBPRD of the EPWM are both 0, EPWMSYNCPER is held at level high and DACxVALA is loaded immediately from DACxVALS irrespective of the value of COMPDACCTL[SWLOADSEL]. Due to this, configure the EPWM first before setting COMPDACCTL[SWLOADSEL] to 1.

---

#### Note

The name of the sync signal that the CMPSS receives from the EPWM has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCL and EPWMSYNCO. For a description of what are these signals, see the *Enhanced Pulse Width Modulator (ePWM)* chapter.

---

### 12.6.2 Synchronizer, Digital Filter, and Latch Delays

The synchronization block adds a delay of 1-2 sysclks. If the digital filter is bypassed (all filter settings are 0), the digital filter adds a delay of 2 sysclks. The latch adds 1 sysclk delay.

### 12.6.3 Calibrating the CMPSS

The CMPSS has two sources of offset errors: comparator offset error and compdac offset error. In the data sheet, the comparator offset error is referred to as **input referred offset error** and compdac offset error is referred to as **Static offset error**. See the device data sheet for the values.

If both inputs of the comparator are driven from a pin, only the comparator offset error applies. However if the inverting input of the comparator is driven from the compdac, then only the compdac offset error applies. This is because the compdac offset error includes comparator offset error.

Due to the offset errors, the CMPSS must be calibrated to make sure trips happen at the expected levels. The following flow outlines how the calibration can be performed if the inverting input of the comparator is driven from the compdac.

Notes before calibration:

1. A static DC signal is required on the non-inverting input of the comparator.
2. Hysteresis can be disabled for calibration and can be re-enabled after calibration is complete.
3. A noisy input can make calibration difficult, so use the latch with non-zero filter settings depending on how noisy is the signal on the non-inverting input.

This approach sweeps down the compdac:

1. Set the starting compdac value to max, 0xFFFF.
  - Optional: Instead of setting the starting compdac value to maximum, set to **Vtarget + Static offset error + Margin**. Where **Vtarget** is the approximate DC voltage on the non-inverting input, **Static offset error** is the compdac offset error specification and **Margin** is some amount of guard band. This can lead to a faster calibration but only works if **Vtarget** is known. Alternatively, if **Vtarget** is unknown, the ADC can be used to convert **Vtarget**.
2. Decrement compdac value by 1.
3. Wait for compdac to settle.
4. Clear latch.
5. Wait for possible latch set.
6. If latch is set, trip code is found exit.
  - Optional: The trip code can be double checked by:
    - a. Increasing compdac value by 1.
    - b. Clear latch.
    - c. Wait for possible latch set.
    - d. Latch can be unset.
7. If latch is unset, go back to step 2 and repeat.

It is also possible to calibrate the CMPSS, if both inputs of the comparator are driven from a pin. For this case, the flow stays the same but the voltage on the inverting pin of the comparator is swept externally.

### 12.6.4 Enabling and Disabling the CMPSS Clock

If the clock to the CMPSS module is disabled while the comparator is active, the following behavior can be expected:

- The comparator remains unaffected and continues to trip from voltages on the inputs.
- If the reference 12-bit DAC is driving the negative input of the comparator, the voltage on the negative input remains static and unaffected but DACVALA can no longer be updated from the ramp generator or DACVALS.
- The ramp generator, synchronize block and digital filter freeze on the current states.

Enabling the clock to the CMPSS restores the clock to the state before the clock was disabled.

## 12.7 CMPSS DAC Output

Select instances of the (full) CMPSS module can optionally route the low DAC output to an external pin to be used as an external DAC. This is subject to the following constraints:

- This is available only on select instances; consult the device data sheet to determine if the CMPSS DAC has been pinned out for a particular instance.
- This functionality is never available on CMPSS\_LITE modules.
- All other CMPSS module functionality is unusable when the output DAC is enabled. This includes the high DAC, both comparators, ramp generation logic, and the digital filters.
- See the device data sheet for effective resolution of the CMPSS DAC output.

---

### Note

The CMPSS low DAC output does not necessarily appear on the same pins that are available for low comparator input. Consult the pinout of the device data sheet to determine which pin the CMPSS output DAC is available on.

---

## 12.8 Software

### 12.8.1 CMPSS Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
 C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/cmpss

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](#).

#### 12.8.1.1 CMPSS Asynchronous Trip

FILE: cmpss\_ex1\_asynch.c

This example enables the CMPSS1 COMPH comparator and feeds the asynchronous CTRIPOUTH signal to the GPIO4/OUTPUTXBAR3 pin and CTRIPH to GPIO13/EPWM7B.

CMPSS is configured to generate trip signals to trip the EPWM signals. CMPIN1P is used to give positive input and internal DAC is configured to provide the negative input. Internal DAC is configured to provide a signal at VDD/2. An EPWM signal is generated at GPIO13 and is configured to be tripped by CTRIPOUTH.

When a low input(VSS) is provided to CMPIN1P,

- Trip signal(GPIO4) output is low
- PWM7B(GPIO13) gives a PWM signal

When a high input(higher than VDD/2) is provided to CMPIN1P,

- Trip signal(GPIO4) output turns high
- PWM7B(GPIO13) gets tripped and outputs as high

#### External Connections

- Give input on CMPIN1P (The pin is shared with ADCINA2)
- Outputs can be observed on GPIO4 and GPIO13 using an oscilloscope

#### Watch Variables

- None

#### 12.8.1.2 CMPSS Digital Filter Configuration

FILE: cmpss\_ex2\_digital\_filter.c

This example enables the CMPSS1 COMPH comparator and feeds the output through the digital filter to the GPIO4/OUTPUTXBAR3 pin.

CMPIN1P is used to give positive input and internal DAC is configured to provide the negative input. Internal DAC is configured to provide a signal at VDD/2.

When a low input(VSS) is provided to CMPIN1P,

- GPIO4 output is low

When a high input(higher than VDD/2) is provided to CMPIN1P,

- GPIO4 output turns high

### 12.8.2 CMPSS\_LITE Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
 C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/cmpss\_lite

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](#).

#### 12.8.2.1 CMPSSLITE Asynchronous Trip

FILE: cmpss\_lite\_ex1\_asynch.c

This example enables the CMPSSLITE2 COMPH comparator and feeds the asynchronous CTRIPOUTH signal to the GPIO4/OUTPUTXBAR3 pin and CTRIPH to GPIO29/EPWM7B.

CMPSS is configured to generate trip signals to trip the EPWM signals. CMPIN2P is used to give positive input and internal DAC is configured to provide the negative input. Internal DAC is configured to provide a signal at VDD/2. An EPWM signal is generated at GPIO29 and is configured to be tripped by CTRIPOUTH.

When a low input(VSS) is provided to CMPIN2P,

- Trip signal(GPIO4) output is low
- PWM7B(GPIO29) gives a PWM signal

When a high input(higher than VDD/2) is provided to CMPIN2P,

- Trip signal(GPIO4) output turns high
- PWM7B(GPIO29) gets tripped and outputs as high

#### External Connections

- Give input on CMPIN2P. The pin is shared with ADCINA9 for CMPHPMXSEL = 2
- Outputs can be observed on GPIO4 and GPIO29 using an oscilloscope

#### Watch Variables

- None

## 12.9 CMPSS Registers

This section describes the CMPSS Registers.

### 12.9.1 CMPSS Base Address Table

**Table 12-2. CMPSS Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
Cmpss1Regs	CMPSS_REGS	CMPSS1_BASE	0x0000_5500	YES
CmpssLite2Regs	CMPSS_LITE_REGS	CMPSSLITE2_BASE	0x0000_5540	YES
CmpssLite3Regs	CMPSS_LITE_REGS	CMPSSLITE3_BASE	0x0000_5580	YES
CmpssLite4Regs	CMPSS_LITE_REGS	CMPSSLITE4_BASE	0x0000_55C0	YES

## 12.9.2 CMPSS\_REGS Registers

Table 12-3 lists the memory-mapped registers for the CMPSS\_REGS registers. All register offset addresses not listed in Table 12-3 should be considered as reserved locations and the register contents should not be modified.

**Table 12-3. CMPSS\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	COMPCTL	CMPSS Comparator Control Register	EALLOW	<a href="#">Go</a>
1h	COMPHYSTL	CMPSS Comparator Hysteresis Control Register	EALLOW	<a href="#">Go</a>
2h	COMPSTS	CMPSS Comparator Status Register		<a href="#">Go</a>
3h	COMPSTSLR	CMPSS Comparator Status Clear Register	EALLOW	<a href="#">Go</a>
4h	COMPDACHCTL	CMPSS High DAC Control Register	EALLOW	<a href="#">Go</a>
6h	DACHVALS	CMPSS High DAC Value Shadow Register		<a href="#">Go</a>
7h	DACHVALA	CMPSS High DAC Value Active Register		<a href="#">Go</a>
8h	RAMPHREFA	CMPSS High Ramp Reference Active Register		<a href="#">Go</a>
Ah	RAMPHREFS	CMPSS High Ramp Reference Shadow Register		<a href="#">Go</a>
Ch	RAMPHSTEPVALA	CMPSS High Ramp Step Value Active Register		<a href="#">Go</a>
Eh	RAMPHSTEPVALS	CMPSS High Ramp Step Value Shadow Register		<a href="#">Go</a>
10h	RAMPHSTS	CMPSS High Ramp Status Register		<a href="#">Go</a>
12h	DACLVALS	CMPSS Low DAC Value Shadow Register		<a href="#">Go</a>
13h	DACLVALA	CMPSS Low DAC Value Active Register		<a href="#">Go</a>
14h	RAMPHDLYA	CMPSS High Ramp Delay Active Register		<a href="#">Go</a>
15h	RAMPHDLYS	CMPSS High Ramp Delay Shadow Register		<a href="#">Go</a>
16h	CTRIPLFILCTL	CTRIPL Filter Control Register	EALLOW	<a href="#">Go</a>
17h	CTRIPLFILCLKCTL	CTRIPL Filter Clock Control Register	EALLOW	<a href="#">Go</a>
18h	CTRIPHFILCTL	CTRIPH Filter Control Register	EALLOW	<a href="#">Go</a>
19h	CTRIPHFILCLKCTL	CTRIPH Filter Clock Control Register	EALLOW	<a href="#">Go</a>
1Ah	COMPLOCK	CMPSS Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 12-4 shows the codes that are used for access types in this section.

**Table 12-4. CMPSS\_REGS Access Type Codes**

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

**Table 12-4. CMPSS\_REGS Access Type Codes (continued)**

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

### 12.9.2.1 COMPCTL Register (Offset = 0h) [Reset = 0000h]

COMPCTL is shown in [Figure 12-8](#) and described in [Table 12-5](#).

Return to the [Summary Table](#).

CMPSS Comparator Control Register

**Figure 12-8. COMPCTL Register**

15	14	13	12	11	10	9	8
COMPDA CE	ASYNCL EN	CTRIPOU TLSEL		CTRIPL SEL		COMPL INV	COMPLSOUR CE
R/W-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED	ASYNCH EN	CTRIPOU THSEL		CTRIPH SEL		COMPH INV	COMPHSOUR CE
R-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h

**Table 12-5. COMPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	COMPDA CE	R/W	0h	Comparator/DAC enable. 0 Comparator/DAC disabled 1 Comparator/DAC enabled Reset type: SYSRSn
14	ASYNCL EN	R/W	0h	Low comparator asynchronous path enable. Allows asynchronous comparator output to feed into OR gate with latched digital filter signal when CTRIPLSEL=3 or CTRIPOUTLSEL=3. 0 Asynchronous comparator output does not feed into OR gate with latched digital filter output 1 Asynchronous comparator output feeds into OR gate with latched digital filter output Reset type: SYSRSn
13-12	CTRIPOU TLSEL	R/W	0h	Low comparator CTRIPOUTL source select. 0 Asynchronous comparator output drives CTRIPOUTL 1 Synchronous comparator output drives CTRIPOUTL 2 Output of digital filter drives CTRIPOUTL 3 Latched output of digital filter drives CTRIPOUTL Reset type: SYSRSn
11-10	CTRIPL SEL	R/W	0h	Low comparator CTRIPL source select. 0 Asynchronous comparator output drives CTRIPL 1 Synchronous comparator output drives CTRIPL 2 Output of digital filter drives CTRIPL 3 Latched output of digital filter drives CTRIPL Reset type: SYSRSn
9	COMPL INV	R/W	0h	Low comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted Reset type: SYSRSn
8	COMPLSOUR CE	R/W	0h	Low comparator input source. 0 Inverting input of comparator driven by internal DAC 1 Inverting input of comparator driven through external pin Reset type: SYSRSn
7	RESERVED	R	0h	Reserved

**Table 12-5. COMPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: SYSRSn
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 Reset type: SYSRSn
3-2	CTRIHSEL	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 Reset type: SYSRSn
1	COMPHINV	R/W	0h	High comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted Reset type: SYSRSn
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 Reset type: SYSRSn

### 12.9.2.2 COMPHYSCTL Register (Offset = 1h) [Reset = 0000h]

COMPHYSCTL is shown in [Figure 12-9](#) and described in [Table 12-6](#).

Return to the [Summary Table](#).

CMPSS Comparator Hysteresis Control Register

**Figure 12-9. COMPHYSCTL 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			

**Table 12-6. COMPHYSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	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 others : undefined Reset type: SYSRSn

### 12.9.2.3 COMPSTS Register (Offset = 2h) [Reset = 0000h]

COMPSTS is shown in [Figure 12-10](#) and described in [Table 12-7](#).

Return to the [Summary Table](#).

CMPSS Comparator Status Register

**Figure 12-10. 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

**Table 12-7. COMPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	COMPLLATCH	R	0h	Latched value of low comparator digital filter output Reset type: SYSRSn
8	COMPLSTS	R	0h	Low comparator digital filter output Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	COMPHLATCH	R	0h	Latched value of high comparator digital filter output Reset type: SYSRSn
0	COMPHSTS	R	0h	High comparator digital filter output Reset type: SYSRSn

### 12.9.2.4 COMPSTSCLR Register (Offset = 3h) [Reset = 0000h]

COMPSTSCLR is shown in [Figure 12-11](#) and described in [Table 12-8](#).

Return to the [Summary Table](#).

CMPSS Comparator Status Clear Register

**Figure 12-11. COMPSTSCLR Register**

15	14	13	12	11	10	9	8
RESERVED					LSYNCCLREN	LLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED					HSYNCCLREN	HLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h

**Table 12-8. COMPSTSCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	LSYNCCLREN	R/W	0h	Low comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. 0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
9	LLATCHCLR	R-0/W1S	0h	Low comparator latch software clear. Perform software reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. Reads always return 0. 0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPLLATCH] Reset type: SYSRSn
8-3	RESERVED	R	0h	Reserved
2	HSYNCCLREN	R/W	0h	High comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. 0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
1	HLATCHCLR	R-0/W1S	0h	High comparator latch software clear. Perform software reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. Reads always return 0. 0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPHLATCH] Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

### 12.9.2.5 COMPDACHCTL Register (Offset = 4h) [Reset = 0000h]

COMPDACHCTL is shown in [Figure 12-12](#) and described in [Table 12-9](#).

Return to the [Summary Table](#).

CMPSS High DAC Control Register

**Figure 12-12. COMPDACHCTL Register**

15	14	13	12	11	10	9	8
FREESOFT		RESERVED	BLANKEN	BLANKSOURCE			
R/W-0h		R-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
SWLOADSEL	RAMPLOADSEL	RESERVED	RAMPSOURCE				DACSOURCE
R/W-0h	R/W-0h	R/W-0h	R/W-0h				R/W-0h

**Table 12-9. COMPDACHCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREESOFT	R/W	0h	Free-run or software-run emulation behavior. Behavior of the ramp generator during emulation suspend. 00b Ramp generator stops immediately during emulation suspend 01b Ramp generator completes current ramp and stops at next EPWMSYNCPER during emulation suspend 1Xb Ramp generator runs freely Reset type: SYSRSn
13	RESERVED	R	0h	Reserved
12	BLANKEN	R/W	0h	COMPH EPWMBLANK enable. This bit enables the EPWMBLANK signal. 0 EPWMBLANK signal is disabled. 1 EPWMBLANK signal is enabled. Reset type: SYSRSn
11-8	BLANKSOURCE	R/W	0h	COMPH EPWMBLANK source select. This bit field determines which EPWMnBLANK is passed on as the EPWMBLANK signal. Where n represents the maximum number of EPWMBLANK signals available on the device: 0 EPWM1BLANK 1 EPWM2BLANK 2 EPWM3BLANK ... n-1 EPWMnBLANK Reset type: SYSRSn
7	SWLOADSEL	R/W	0h	Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or EPWMSYNCPER. 0 DACxVALA is updated from DACxVALS on SYSCLK 1 DACxVALA is updated from DACxVALS on EPWMSYNCPER Reset type: SYSRSn
6	RAMPLOADSEL	R/W	0h	Ramp load select. Determines whether RAMPHSTS is updated from RAMPHREFA or RAMPHREFS when COMPSTS[COMPHSTS] is triggered. 0 RAMPHSTS is loaded from RAMPHREFA 1 RAMPHSTS is loaded from RAMPHREFS Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved

**Table 12-9. COMPDACHCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-1	RAMPSOURCE	R/W	0h	High Ramp generator source select. Determines which EPWMSYNCPER signal is used within the COMPH Where n represents the maximum number of EPWMSYNCPER signals available on the device: 0 EPWM1SYNCPER 1 EPWM2SYNCPER 2 EPWM3SYNCPER ... n-1 EPWMnSYNCPER Reset type: SYSRSn
0	DACSOURCE	R/W	0h	DACH source select. Determines whether DACHVALA is updated from DACHVALS or from the high ramp generator. 0 DACHVALA is updated from DACHVALS 1 DACHVALA is updated from the high ramp generator Reset type: SYSRSn

### 12.9.2.6 DACHVALS Register (Offset = 6h) [Reset = 0000h]

DACHVALS is shown in [Figure 12-13](#) and described in [Table 12-10](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Shadow Register

**Figure 12-13. DACHVALS Register**

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

**Table 12-10. DACHVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R/W	0h	High DAC shadow value. When COMPDACCTL[DACSOURCE]=0, the value of DACHVALS is loaded into DACHVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

### 12.9.2.7 DACHVALA Register (Offset = 7h) [Reset = 0000h]

DACHVALA is shown in [Figure 12-14](#) and described in [Table 12-11](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Active Register

**Figure 12-14. DACHVALA Register**

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

**Table 12-11. DACHVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	High DAC active value. Value that is actively driven by the high DAC. Reset type: SYSRSn

### 12.9.2.8 RAMPHREFA Register (Offset = 8h) [Reset = 0000h]

RAMPHREFA is shown in [Figure 12-15](#) and described in [Table 12-12](#).

Return to the [Summary Table](#).

CMPSS High Ramp Reference Active Register

**Figure 12-15. RAMPHREFA Register**

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

**Table 12-12. RAMPHREFA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPHREF	R	0h	High Ramp reference active value. Latched value to be loaded into ramp generator RAMHPSTS. Reset type: SYSRSn

### 12.9.2.9 RAMPHREFS Register (Offset = Ah) [Reset = 0000h]

RAMPHREFS is shown in [Figure 12-16](#) and described in [Table 12-13](#).

Return to the [Summary Table](#).

CMPSS High Ramp Reference Shadow Register

**Figure 12-16. RAMPHREFS Register**

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

**Table 12-13. RAMPHREFS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPHREF	R/W	0h	High Ramp reference shadow. Unlatched value to be loaded into ramp generator RAMPHSTS. Reset type: SYSRSn

### 12.9.2.10 RAMPHSTEPVALA Register (Offset = Ch) [Reset = 0000h]

RAMPHSTEPVALA is shown in [Figure 12-17](#) and described in [Table 12-14](#).

Return to the [Summary Table](#).

CMPSS High Ramp Step Value Active Register

**Figure 12-17. RAMPHSTEPVALA Register**

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

**Table 12-14. RAMPHSTEPVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPHSTEPVAL	R	0h	High Ramp step value active. Latched value that will be subtracted from RAMPHSTS. Reset type: SYSRSn

### 12.9.2.11 RAMPHSTEPVALS Register (Offset = Eh) [Reset = 0000h]

RAMPHSTEPVALS is shown in [Figure 12-18](#) and described in [Table 12-15](#).

Return to the [Summary Table](#).

CMPSS High Ramp Step Value Shadow Register

**Figure 12-18. RAMPHSTEPVALS Register**

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

**Table 12-15. RAMPHSTEPVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPHSTEPVAL	R/W	0h	High Ramp step value shadow. Unlatched value to be loaded into RAMPHSTEPVALA. Reset type: SYSRSn

### 12.9.2.12 RAMPHSTS Register (Offset = 10h) [Reset = 0000h]

RAMPHSTS is shown in [Figure 12-19](#) and described in [Table 12-16](#).

Return to the [Summary Table](#).

CMPSS High Ramp Status Register

**Figure 12-19. RAMPHSTS Register**

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

**Table 12-16. RAMPHSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RAMPHVALUE	R	0h	High Ramp value. Present value of ramp generator. Reset type: SYSRSn

### 12.9.2.13 DACLVALS Register (Offset = 12h) [Reset = 0000h]

DACLVALS is shown in [Figure 12-20](#) and described in [Table 12-17](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Shadow Register

**Figure 12-20. DACLVALS Register**

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

**Table 12-17. 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 DACVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

### 12.9.2.14 DACLVALA Register (Offset = 13h) [Reset = 0000h]

DACLVALA is shown in [Figure 12-21](#) and described in [Table 12-18](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Active Register

**Figure 12-21. DACLVALA Register**

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

**Table 12-18. DACLVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	Low DAC active value. Value that is actively driven by the low DAC. Reset type: SYSRSn

### 12.9.2.15 RAMPHDLYA Register (Offset = 14h) [Reset = 0000h]

RAMPHDLYA is shown in [Figure 12-22](#) and described in [Table 12-19](#).

Return to the [Summary Table](#).

CMPSS High Ramp Delay Active Register

**Figure 12-22. RAMPHDLYA Register**

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

**Table 12-19. RAMPHDLYA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-0	DELAY	R	0h	High Ramp delay active value. Latched value of the number of cycles to delay the start of the ramp generator stepper after a EPWMSYNCPER is received. Reset type: SYSRSn

### 12.9.2.16 RAMPHDLYS Register (Offset = 15h) [Reset = 0000h]

RAMPHDLYS is shown in [Figure 12-23](#) and described in [Table 12-20](#).

Return to the [Summary Table](#).

CMPSS High Ramp Delay Shadow Register

**Figure 12-23. RAMPHDLYS Register**

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

**Table 12-20. RAMPHDLYS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-0	DELAY	R/W	0h	High Ramp delay shadow value. Unlatched value to be loaded into RAMPHDLYA. Reset type: SYSRSn

### 12.9.2.17 CTRIPLFILCTL Register (Offset = 16h) [Reset = 0000h]

CTRIPLFILCTL is shown in [Figure 12-24](#) and described in [Table 12-21](#).

Return to the [Summary Table](#).

CTRIPL Filter Control Register

**Figure 12-24. CTRIPLFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				RESERVED	
R-0/W1S-0h	R-0h	R/W-0h				R-0h	
7	6	5	4	3	2	1	0
SAMPWIN					RESERVED		
R/W-0h					R-0h		

**Table 12-21. CTRIPLFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	Low filter initialization. 0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	Low filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-3	SAMPWIN	R/W	0h	Low filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 12.9.2.18 CTRIPLFILCLKCTL Register (Offset = 17h) [Reset = 0000h]

CTRIPLFILCLKCTL is shown in [Figure 12-25](#) and described in [Table 12-22](#).

Return to the [Summary Table](#).

CTRIPL Filter Clock Control Register

**Figure 12-25. CTRIPLFILCLKCTL Register**

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

**Table 12-22. CTRIPLFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CLKPRESCALE	R/W	0h	Low filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

### 12.9.2.19 CTRIPFILCTL Register (Offset = 18h) [Reset = 0000h]

CTRIPFILCTL is shown in [Figure 12-26](#) and described in [Table 12-23](#).

Return to the [Summary Table](#).

CTRIPH Filter Control Register

**Figure 12-26. CTRIPFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				RESERVED	
R-0/W1S-0h	R-0h	R/W-0h				R-0h	
7	6	5	4	3	2	1	0
SAMPWIN				RESERVED			
R/W-0h				R-0h			

**Table 12-23. CTRIPFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	High filter initialization. 0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	High filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-3	SAMPWIN	R/W	0h	High filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 12.9.2.20 CTRIPFILCLKCTL Register (Offset = 19h) [Reset = 0000h]

CTRIPFILCLKCTL is shown in [Figure 12-27](#) and described in [Table 12-24](#).

Return to the [Summary Table](#).

CTRIPH Filter Clock Control Register

**Figure 12-27. CTRIPFILCLKCTL Register**

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

**Table 12-24. CTRIPFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CLKPRESCALE	R/W	0h	High filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

### 12.9.2.21 COMPLOCK Register (Offset = 1Ah) [Reset = 0000h]

COMPLOCK is shown in [Figure 12-28](#) and described in [Table 12-25](#).

Return to the [Summary Table](#).

CMPSS Lock Register

**Figure 12-28. 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	COMPHYSTL	COMPCTL
R-0h			R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 12-25. COMPLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	CTRIIP	R/WOnce	0h	Lock write-access to the CTRIPxFILTCTL and CTRIPxFILCLKCTL* registers. 0 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are not locked. Write 0 to this bit has no effect. 1 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are locked. Only a system reset can clear this bit. Reset type: SYSRSn
2	DACCTL	R/WOnce	0h	Lock write-access to the COMPDAC*CTL* register(s). 0 COMPDAC*CTL* register(s) not locked. Write 0 to this bit has no effect. 1 COMPDAC*CTL* register(s) locked. Only a system reset can clear this bit. Reset type: SYSRSn
1	COMPHYSTL	R/WOnce	0h	Lock write-access to the COMPHYSTL register. 0 COMPHYSTL register is not locked. Write 0 to this bit has no effect. 1 COMPHYSTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
0	COMPCTL	R/WOnce	0h	Lock write-access to the COMPCTL register. 0 COMPCTL register is not locked. Write 0 to this bit has no effect. 1 COMPCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn

### 12.9.3 CMPSS\_LITE\_REGS Registers

Table 12-26 lists the memory-mapped registers for the CMPSS\_LITE\_REGS registers. All register offset addresses not listed in Table 12-26 should be considered as reserved locations and the register contents should not be modified.

**Table 12-26. CMPSS\_LITE\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	COMPCTL	CMPSS Comparator Control Register	EALLOW	<a href="#">Go</a>
1h	COMPHYSTL	CMPSS Comparator Hysteresis Control Register	EALLOW	<a href="#">Go</a>
2h	COMPSTS	CMPSS Comparator Status Register		<a href="#">Go</a>
3h	COMPSTSLR	CMPSS Comparator Status Clear Register	EALLOW	<a href="#">Go</a>
4h	COMPDACHCTL	CMPSS High DAC Control Register	EALLOW	<a href="#">Go</a>
6h	DACHVALS	CMPSS High DAC Value Shadow Register		<a href="#">Go</a>
7h	DACHVALA	CMPSS High DAC Value Active Register		<a href="#">Go</a>
12h	DACLVALS	CMPSS Low DAC Value Shadow Register		<a href="#">Go</a>
13h	DACLVALA	CMPSS Low DAC Value Active Register		<a href="#">Go</a>
16h	CTRIPLFILCTL	CTRIPL Filter Control Register	EALLOW	<a href="#">Go</a>
17h	CTRIPLFILCLKCTL	CTRIPL Filter Clock Control Register	EALLOW	<a href="#">Go</a>
18h	CTRIPHFILCTL	CTRIPH Filter Control Register	EALLOW	<a href="#">Go</a>
19h	CTRIPHFILCLKCTL	CTRIPH Filter Clock Control Register	EALLOW	<a href="#">Go</a>
1Ah	COMPLOCK	CMPSS Lock Register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 12-27 shows the codes that are used for access types in this section.

**Table 12-27. CMPSS\_LITE\_REGS Access Type Codes**

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

### 12.9.3.1 COMPCTL Register (Offset = 0h) [Reset = 0000h]

COMPCTL is shown in [Figure 12-29](#) and described in [Table 12-28](#).

Return to the [Summary Table](#).

CMPSS Comparator Control Register

**Figure 12-29. COMPCTL Register**

15	14	13	12	11	10	9	8
COMPDA CE	ASYN CLEN	CTRIP OUTLSEL		CTRIP LSEL		COMPL INV	COMPL SOURC E
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	ASYN CHEN	CTRIP OUTHSEL		CTRIP HSEL		COMPH INV	COMPH SOURC E
R-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h

**Table 12-28. COMPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	COMPDA CE	R/W	0h	Comparator/DAC enable. 0 Comparator/DAC disabled 1 Comparator/DAC enabled Reset type: SYSRSn
14	ASYN CLEN	R/W	0h	Low comparator asynchronous path enable. Allows asynchronous comparator output to feed into OR gate with latched digital filter signal when CTRIP LSEL=3 or CTRIP OUTHSEL=3. 0 Asynchronous comparator output does not feed into OR gate with latched digital filter output 1 Asynchronous comparator output feeds into OR gate with latched digital filter output Reset type: SYSRSn
13-12	CTRIP OUTHSEL	R/W	0h	Low comparator CTRIP OUTL source select. 0 Asynchronous comparator output drives CTRIP OUTL 1 Synchronous comparator output drives CTRIP OUTL 2 Output of digital filter drives CTRIP OUTL 3 Latched output of digital filter drives CTRIP OUTL Reset type: SYSRSn
11-10	CTRIP LSEL	R/W	0h	Low comparator CTRIP L source select. 0 Asynchronous comparator output drives CTRIP L 1 Synchronous comparator output drives CTRIP L 2 Output of digital filter drives CTRIP L 3 Latched output of digital filter drives CTRIP L Reset type: SYSRSn
9	COMPL INV	R/W	0h	Low comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted Reset type: SYSRSn
8	COMPL SOURCE	R/W	0h	Low comparator input source. 0 Inverting input of comparator driven by internal DAC 1 Inverting input of comparator driven through external pin Reset type: SYSRSn
7	RESERVED	R	0h	Reserved

**Table 12-28. COMPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 Reset type: SYSRSn
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 Reset type: SYSRSn
3-2	CTRIHSEL	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 Reset type: SYSRSn
1	COMPHINV	R/W	0h	High comparator output invert. 0 Output of comparator is not inverted 1 Output of comparator is inverted Reset type: SYSRSn
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 Reset type: SYSRSn

### 12.9.3.2 COMPHYSCTL Register (Offset = 1h) [Reset = 0000h]

COMPHYSCTL is shown in [Figure 12-30](#) and described in [Table 12-29](#).

Return to the [Summary Table](#).

CMPSS Comparator Hysteresis Control Register

**Figure 12-30. COMPHYSCTL 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			

**Table 12-29. COMPHYSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-0	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 others : undefined Reset type: SYSRSn

### 12.9.3.3 COMPSTS Register (Offset = 2h) [Reset = 0000h]

COMPSTS is shown in [Figure 12-31](#) and described in [Table 12-30](#).

Return to the [Summary Table](#).

CMPSS Comparator Status Register

**Figure 12-31. 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

**Table 12-30. COMPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	COMPLLATCH	R	0h	Latched value of low comparator digital filter output Reset type: SYSRSn
8	COMPLSTS	R	0h	Low comparator digital filter output Reset type: SYSRSn
7-2	RESERVED	R	0h	Reserved
1	COMPHLATCH	R	0h	Latched value of high comparator digital filter output Reset type: SYSRSn
0	COMPHSTS	R	0h	High comparator digital filter output Reset type: SYSRSn

### 12.9.3.4 COMPSTSCLR Register (Offset = 3h) [Reset = 0000h]

COMPSTSCLR is shown in [Figure 12-32](#) and described in [Table 12-31](#).

Return to the [Summary Table](#).

CMPSS Comparator Status Clear Register

**Figure 12-32. COMPSTSCLR Register**

15	14	13	12	11	10	9	8
RESERVED					LSYNCCLREN	LLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED					HSYNCCLREN	HLATCHCLR	RESERVED
R-0h					R/W-0h	R-0/W1S-0h	R-0h

**Table 12-31. COMPSTSCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	LSYNCCLREN	R/W	0h	Low comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. 0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
9	LLATCHCLR	R-0/W1S	0h	Low comparator latch software clear. Perform software reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. Reads always return 0. 0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPLLATCH] Reset type: SYSRSn
8-3	RESERVED	R	0h	Reserved
2	HSYNCCLREN	R/W	0h	High comparator latch EPWMSYNCPER clear. Enable EPWMSYNCPER reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. 0 EPWMSYNCPER will not reset latch 1 EPWMSYNCPER will reset latch Reset type: SYSRSn
1	HLATCHCLR	R-0/W1S	0h	High comparator latch software clear. Perform software reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. Reads always return 0. 0 No effect 1 Generate a single pulse of latch reset signal for COMPSTS[COMPHLATCH] Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

### 12.9.3.5 COMPDACHCTL Register (Offset = 4h) [Reset = 0000h]

COMPDACHCTL is shown in [Figure 12-33](#) and described in [Table 12-32](#).

Return to the [Summary Table](#).

CMPSS High DAC Control Register

**Figure 12-33. COMPDACHCTL Register**

15	14	13	12	11	10	9	8
RESERVED		RESERVED	BLANKEN	BLANKSOURCE			
R/W-0h		R-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
SWLOADSEL	RESERVED	RESERVED	RAMPSOURCE			RESERVED	
R/W-0h	R/W-0h	R/W-0h	R/W-0h			R/W-0h	

**Table 12-32. COMPDACHCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R/W	0h	Reserved
13	RESERVED	R	0h	Reserved
12	BLANKEN	R/W	0h	COMPH EPWMBLANK enable. This bit enables the EPWMBLANK signal. 0 EPWMBLANK signal is disabled. 1 EPWMBLANK signal is enabled. Reset type: SYSRSn
11-8	BLANKSOURCE	R/W	0h	COMPH EPWMBLANK source select. This bit field determines which EPWmBLANK is passed on as the EPWMBLANK signal. Where n represents the maximum number of EPWMBLANK signals available on the device: 0 EPWM1BLANK 1 EPWM2BLANK 2 EPWM3BLANK ... n-1 EPWmBLANK Reset type: SYSRSn
7	SWLOADSEL	R/W	0h	Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or EPWMSYNCPER. 0 DACxVALA is updated from DACxVALS on SYSCLK 1 DACxVALA is updated from DACxVALS on EPWMSYNCPER Reset type: SYSRSn
6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4-1	RAMPSOURCE	R/W	0h	EPWMSYNCPER source select. Determines which EPWmSYNCPER signal is used within the COMPH Where n represents the maximum number of EPWMSYNCPER signals available on the device: 0 EPWM1SYNCPER 1 EPWM2SYNCPER 2 EPWM3SYNCPER ... n-1 EPWmSYNCPER Reset type: SYSRSn
0	RESERVED	R/W	0h	Reserved

### 12.9.3.6 DACHVALS Register (Offset = 6h) [Reset = 0000h]

DACHVALS is shown in [Figure 12-34](#) and described in [Table 12-33](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Shadow Register

**Figure 12-34. DACHVALS Register**

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

**Table 12-33. DACHVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R/W	0h	High DAC shadow value. When COMPDACCTL[DACSOURCE]=0, the value of DACHVALS is loaded into DACHVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

### 12.9.3.7 DACHVALA Register (Offset = 7h) [Reset = 0000h]

DACHVALA is shown in [Figure 12-35](#) and described in [Table 12-34](#).

Return to the [Summary Table](#).

CMPSS High DAC Value Active Register

**Figure 12-35. DACHVALA Register**

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

**Table 12-34. DACHVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	High DAC active value. Value that is actively driven by the high DAC. Reset type: SYSRSn

### 12.9.3.8 DACLVALS Register (Offset = 12h) [Reset = 0000h]

DACLVALS is shown in [Figure 12-36](#) and described in [Table 12-35](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Shadow Register

**Figure 12-36. DACLVALS Register**

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

**Table 12-35. DACLVALS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACLVAL	R/W	0h	Low DAC shadow value. value to be loaded into DACLVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL]. Reset type: SYSRSn

### 12.9.3.9 DACLVALA Register (Offset = 13h) [Reset = 0000h]

DACLVALA is shown in [Figure 12-37](#) and described in [Table 12-36](#).

Return to the [Summary Table](#).

CMPSS Low DAC Value Active Register

**Figure 12-37. DACLVALA Register**

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

**Table 12-36. DACLVALA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-0	DACVAL	R	0h	Low DAC active value. Value that is actively driven by the low DAC. Reset type: SYSRSn

### 12.9.3.10 CTRIPLFILCTL Register (Offset = 16h) [Reset = 0000h]

CTRIPLFILCTL is shown in [Figure 12-38](#) and described in [Table 12-37](#).

Return to the [Summary Table](#).

CTRIPL Filter Control Register

**Figure 12-38. CTRIPLFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				RESERVED	
R-0/W1S-0h	R-0h	R/W-0h				R-0h	
7	6	5	4	3	2	1	0
SAMPWIN					RESERVED		
R/W-0h					R-0h		

**Table 12-37. CTRIPLFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	Low filter initialization. 0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	Low filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-3	SAMPWIN	R/W	0h	Low filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 12.9.3.11 CTRIPLFILCLKCTL Register (Offset = 17h) [Reset = 0000h]

CTRIPLFILCLKCTL is shown in [Figure 12-39](#) and described in [Table 12-38](#).

Return to the [Summary Table](#).

CTRIPL Filter Clock Control Register

**Figure 12-39. CTRIPLFILCLKCTL Register**

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

**Table 12-38. CTRIPLFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CLKPRESCALE	R/W	0h	Low filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

### 12.9.3.12 CTRIPFILCTL Register (Offset = 18h) [Reset = 0000h]

CTRIPFILCTL is shown in [Figure 12-40](#) and described in [Table 12-39](#).

Return to the [Summary Table](#).

CTRIPH Filter Control Register

**Figure 12-40. CTRIPFILCTL Register**

15	14	13	12	11	10	9	8
FILINIT	RESERVED	THRESH				RESERVED	
R-0/W1S-0h	R-0h	R/W-0h				R-0h	
7	6	5	4	3	2	1	0
SAMPWIN					RESERVED		
R/W-0h					R-0h		

**Table 12-39. CTRIPFILCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	FILINIT	R-0/W1S	0h	High filter initialization. 0 No effect 1 Initialize all samples to the filter input value Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-9	THRESH	R/W	0h	High filter majority voting threshold. At least THRESH samples of the opposite state must appear within the sample window in order for the output to change state. Threshold used is THRESH+1. Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-3	SAMPWIN	R/W	0h	High filter sample window size. Number of samples to monitor is SAMPWIN+1. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 12.9.3.13 CTRIPFILCLKCTL Register (Offset = 19h) [Reset = 0000h]

CTRIPFILCLKCTL is shown in [Figure 12-41](#) and described in [Table 12-40](#).

Return to the [Summary Table](#).

CTRIPH Filter Clock Control Register

**Figure 12-41. CTRIPFILCLKCTL Register**

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

**Table 12-40. CTRIPFILCLKCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CLKPRESCALE	R/W	0h	High filter sample clock prescale. Number of system clocks between samples is CLKPRESCALE+1. Reset type: SYSRSn

### 12.9.3.14 COMPLOCK Register (Offset = 1Ah) [Reset = 0000h]

COMPLOCK is shown in [Figure 12-42](#) and described in [Table 12-41](#).

Return to the [Summary Table](#).

CMPSS Lock Register

**Figure 12-42. 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	COMPHYSTL	COMPCTL	
R-0h		R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	

**Table 12-41. COMPLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	RESERVED	R/WOnce	0h	Reserved
3	CTRIIP	R/WOnce	0h	Lock write-access to the CTRIPxFILTCTL and CTRIPxFILCLKCTL* registers. 0 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are not locked. Write 0 to this bit has no effect. 1 CTRIPxFILCTL and CTRIPxFILCLKCTL* registers are locked. Only a system reset can clear this bit. Reset type: SYSRSn
2	DACCTL	R/WOnce	0h	Lock write-access to the COMPDAC*CTL* register(s). 0 COMPDAC*CTL* register(s) not locked. Write 0 to this bit has no effect. 1 COMPDAC*CTL* register(s) locked. Only a system reset can clear this bit. Reset type: SYSRSn
1	COMPHYSTL	R/WOnce	0h	Lock write-access to the COMPHYSTL register. 0 COMPHYSTL register is not locked. Write 0 to this bit has no effect. 1 COMPHYSTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn
0	COMPCTL	R/WOnce	0h	Lock write-access to the COMPCTL register. 0 COMPCTL register is not locked. Write 0 to this bit has no effect. 1 COMPCTL register is locked. Only a system reset can clear this bit. Reset type: SYSRSn

### 12.9.4 CMPSS Registers to Driverlib Functions

**Table 12-42. CMPSS Registers to Driverlib Functions**

File	Driverlib Function
<b>COMPCTL</b>	
cmpss.h	CMPSS_enableModule
cmpss.h	CMPSS_disableModule
cmpss.h	CMPSS_configHighComparator
cmpss.h	CMPSS_configLowComparator
cmpss.h	CMPSS_configOutputsHigh
cmpss.h	CMPSS_configOutputsLow

**Table 12-42. CMPSS Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>COMPHYSCTL</b>	
cmpss.h	CMPSS_setHysteresis
<b>COMPSTS</b>	
cmpss.c	CMPSS_configLatchOnPWMSYNC
cmpss.h	CMPSS_getStatus
cmpss.h	CMPSS_clearFilterLatchHigh
cmpss.h	CMPSS_clearFilterLatchLow
cmpss.h	CMPSS_enableLatchResetOnPWMSYNCHigh
cmpss.h	CMPSS_disableLatchResetOnPWMSYNCHigh
cmpss.h	CMPSS_enableLatchResetOnPWMSYNCLow
cmpss.h	CMPSS_disableLatchResetOnPWMSYNCLow
<b>COMPSTCLR</b>	
cmpss.c	CMPSS_configLatchOnPWMSYNC
cmpss.h	CMPSS_clearFilterLatchHigh
cmpss.h	CMPSS_clearFilterLatchLow
cmpss.h	CMPSS_enableLatchResetOnPWMSYNCHigh
cmpss.h	CMPSS_disableLatchResetOnPWMSYNCHigh
cmpss.h	CMPSS_enableLatchResetOnPWMSYNCLow
cmpss.h	CMPSS_disableLatchResetOnPWMSYNCLow
<b>COMPDACHCTL</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_configDAC
cmpss.h	CMPSS_configureSyncSourceHigh
cmpss.h	CMPSS_configBlanking
cmpss.h	CMPSS_enableBlanking
cmpss.h	CMPSS_disableBlanking
<b>DACHVALS</b>	
cmpss.h	CMPSS_setDACValueHigh
<b>DACHVALA</b>	
cmpss.h	CMPSS_getDACValueHigh
<b>RAMPHREFA</b>	
cmpss.h	CMPSS_getMaxRampValue
<b>RAMPHREFS</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_setMaxRampValue
<b>RAMPHSTEPVALA</b>	
cmpss.h	CMPSS_getRampDecValue
<b>RAMPHSTEPVALS</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_setRampDecValue
<b>RAMPHSTS</b>	
-	
<b>DACLVALS</b>	
cmpss.h	CMPSS_setDACValueLow
<b>DACLVALA</b>	

**Table 12-42. CMPSS Registers to Driverlib Functions (continued)**

File	Driverlib Function
cmpss.h	CMPSS_getDACValueLow
<b>RAMPHDLYA</b>	
cmpss.h	CMPSS_getRampDelayValue
<b>RAMPHDLYS</b>	
cmpss.c	CMPSS_configRamp
cmpss.h	CMPSS_setRampDelayValue
<b>CTRIPLFILCTL</b>	
cmpss.c	CMPSS_configFilterLow
cmpss.h	CMPSS_initFilterLow
<b>CTRIPLFILCLKCTL</b>	
cmpss.c	CMPSS_configFilterLow
<b>CTRIPHFILCTL</b>	
cmpss.c	CMPSS_configFilterHigh
cmpss.h	CMPSS_initFilterHigh
<b>CTRIPHFILCLKCTL</b>	
cmpss.c	CMPSS_configFilterHigh
<b>COMPLOCK</b>	
-	

### 12.9.5 CMPSS\_LITE Registers to Driverlib Functions

**Table 12-43. CMPSS\_LITE Registers to Driverlib Functions**

File	Driverlib Function
<b>COMPCTL</b>	
-	
<b>COMPHYSCTL</b>	
-	
<b>COMPSTS</b>	
-	
<b>COMPSTSCLR</b>	
-	
<b>COMPDACHCTL</b>	
cmpss_lite.h	staticinlinevoidCMPSSLITE_configDAC
<b>DACHVALS</b>	
-	
<b>DACHVALA</b>	
-	
<b>DACLVALS</b>	
-	
<b>DACLVALA</b>	
-	
<b>CTRIPLFILCTL</b>	
-	
<b>CTRIPLFILCLKCTL</b>	
-	
<b>CTRIPHFILCTL</b>	

**Table 12-43. CMPSS\_LITE Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	
<b>CTRIPHFILCLKCTL</b>	
-	
<b>COMPLOCK</b>	
-	

## Chapter 13 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 can also perform a digital-to-analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a power DAC.

This chapter is applicable for ePWM type 4 with added register protection capability. See the [C2000 Real-Time Control Peripheral Reference Guide](#) for a list of all devices with an ePWM module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

<b>13.1 Introduction</b> .....	<b>1244</b>
<b>13.2 Configuring Device Pins</b> .....	<b>1251</b>
<b>13.3 ePWM Modules Overview</b> .....	<b>1251</b>
<b>13.4 Time-Base (TB) Submodule</b> .....	<b>1253</b>
<b>13.5 Counter-Compare (CC) Submodule</b> .....	<b>1268</b>
<b>13.6 Action-Qualifier (AQ) Submodule</b> .....	<b>1274</b>
<b>13.7 Dead-Band Generator (DB) Submodule</b> .....	<b>1287</b>
<b>13.8 PWM Chopper (PC) Submodule</b> .....	<b>1294</b>
<b>13.9 Trip-Zone (TZ) Submodule</b> .....	<b>1298</b>
<b>13.10 Event-Trigger (ET) Submodule</b> .....	<b>1304</b>
<b>13.11 Digital Compare (DC) Submodule</b> .....	<b>1309</b>
<b>13.12 ePWM Crossbar (X-BAR)</b> .....	<b>1319</b>
<b>13.13 Applications to Power Topologies</b> .....	<b>1320</b>
<b>13.14 Register Lock Protection</b> .....	<b>1338</b>
<b>13.15 High-Resolution Pulse Width Modulator (HRPWM)</b> .....	<b>1339</b>
<b>13.16 Software</b> .....	<b>1365</b>
<b>13.17 ePWM Registers</b> .....	<b>1372</b>

## 13.1 Introduction

This chapter includes an overview and information about each submodule:

- [Time Base \(TB\) Submodule](#)
- [Counter Compare \(CC\) Submodule](#)
- [Action Qualifier \(AQ\) Submodule](#)
- [Dead-Band Generator \(DB\) Submodule](#)
- [PWM Chopper \(PC\) Submodule](#)
- [Trip Zone \(TZ\) Submodule](#)
- [Event Trigger \(ET\) Submodule](#)
- [Digital Compare \(DC\) Submodule](#)

The ePWM Type 4 is functionally compatible to Type 2 (a Type 3 does not exist). Type 4 has the following enhancements in addition to the Type 2 features:

- **Register Address Map:** Additional registers are required for new features on ePWM Type 4. The ePWM register address space has been remapped for better alignment and easy usage.
- **Delayed Trip Functionality:** Changes have been added to achieve deadband insertion capabilities to support, for example, delayed trip functionality needed for peak current mode control type application scenarios. This has been accomplished by allowing comparator events to go into the Action Qualifier as a trigger event (Events T1 and T2). If comparator T1 / T2 events are used to edit the PWM, changes to the PWM waveform do not take place immediately. Instead, the waveform synchronizes to the next TBCLK.
- **Dead-Band Generator Submodule Enhancements:** Shadowing of the DBCTL register to allow dynamic configuration changes.
- **One Shot and Global Load of Registers:** The ePWM Type 4 allows one shot and global load capability from shadow to active registers to avoid partial loads in, for example, multiphase applications. ePWM Type 4 also allows a programmable prescale of shadow to active load events. ePWM Type 4 Global Load can simplify ePWM software by removing interrupts and making sure that all registers are loaded at the same time.
- **Trip-Zone Submodule Enhancements:** Independent flags have been added to reflect the trip status for each of the TZ sources. Changes have been made to the trip-zone submodule to support certain power converter switching techniques like valley switching.
- **Digital Compare Submodule Enhancements:** Blanking window filter register width has been increased from 8 to 16 bits. DCCAP functionality has been enhanced to provide more programmability.
- **PWM SYNC Related Enhancements:** The ePWM Type 4 allows PWM SYNCOUT generation based on CMPC and CMPD events. These events can also be used for PWMSYNC pulse selection.

The ePWM Type 2 is fully compatible to Type 1. Type 2 has the following enhancements in addition to the Type 1 features:

- **High-Resolution Dead-Band Capability:** High-resolution capability is added to dead-band RED and FED in half-cycle clocking mode.
- **Dead-Band Generator Submodule Enhancements:** The ePWM Type 2 has features to enable both RED and FED on either PWM outputs. Provides increased dead band with 14-bit counters and dead-band / dead-band high-resolution registers are shadowed
- **High-Resolution Extension available on ePWMxB outputs:** Provides the ability to enable high-resolution period and duty cycle control on ePWMxB outputs. This is discussed in more detail in [Section 13.15](#).
- **Counter Compare Submodule Enhancements:** The ePWM Type 2 allows interrupts and SOC events to be generated by additional counter compares CMPC and CMPD.
- **Event Trigger Submodule Enhancements:** Prescaling logic to issue interrupt requests and ADC start of conversion expanded up to every 15 events. This submodule allows software initialization of event counters on SYNC event.
- **Digital Compare Submodule Enhancements:** Digital Compare Trip Select logic [DCTRIPSEL] 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 and must be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel submodules with separate resources that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand the operation quickly.

In this document, the letter x within a signal or submodule name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

### Type0 to Type1 Enhancements

- **Increased Dead-Band Resolution:** Dead-band clocking has been enhanced to allow half-cycle clocking to double resolution.
- **Enhanced Interrupt and SOC Generation:** Interrupts and ADC start-of-conversion can now be generated on both the TBCTR == zero and TBCTR == period events. This feature enables dual edge PWM control. Additionally, the ADC start-of-conversion can be generated from an event defined in the digital compare submodule.
- **High-Resolution Period Capability:** Provides the ability to enable high-resolution period. This is discussed in more detail in [Section 13.15](#).
- **Digital Compare Submodule:** The digital compare submodule enhances the event triggering and trip zone submodules by providing filtering, blanking and improved trip functionality to digital compare signals. Such features are essential for peak current mode control and for support of analog comparators.

---

#### Note

The name of the sync signal that goes to the CMPSS has been updated from PWMSYNC to EPWMSYNCPER (SYNCPER/PWMSYNCPER/EPWMxSYNCPER) to avoid confusion with the other EPWM sync signals EPWMSYNCI and EPWMSYNCO. For a description of these signals, see [Table 13-2](#).

---

### 13.1.1 EPWM Related Collateral

#### Foundational Materials

- [C2000 Academy - EPWM](#)
- [Real-Time Control Reference Guide](#)
  - Refer to the EPWM section

#### Getting Started Materials

- [C2000 ePWM Developer's Guide Application Report](#)
- [Enhanced Pulse Width Modulator \(ePWM\) Training for C2000 MCUs \(Video\)](#)
- [Flexible PWMs Enable Multi-Axis Drives, Multi-Level Inverters Application Report](#)
- [Getting Started with the C2000 ePWM Module \(Video\)](#)
- [Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Control Application Report](#)

- Chapters 1 to 6 are Fundamental material, derivations, and explanations that are useful for learning about how PWM can be used to implement a DAC. Subsequent chapters are Getting Started and Expert material for implementing in a system.
- [Using the Enhanced Pulse Width Modulator \(ePWM\) Module Application Report](#)

### Expert Materials

- [C2000 real-time microcontrollers - Reference designs](#)
  - See TI designs related to specific end applications used.
- [CRM/ZVS PFC Implementation Based on C2000 Type-4 PWM Module Application Report](#)
- [Leverage New Type ePWM Features for Multiple Phase Control Application Report](#)

### 13.1.2 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 [Section 13.15](#). See the device data sheet 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.

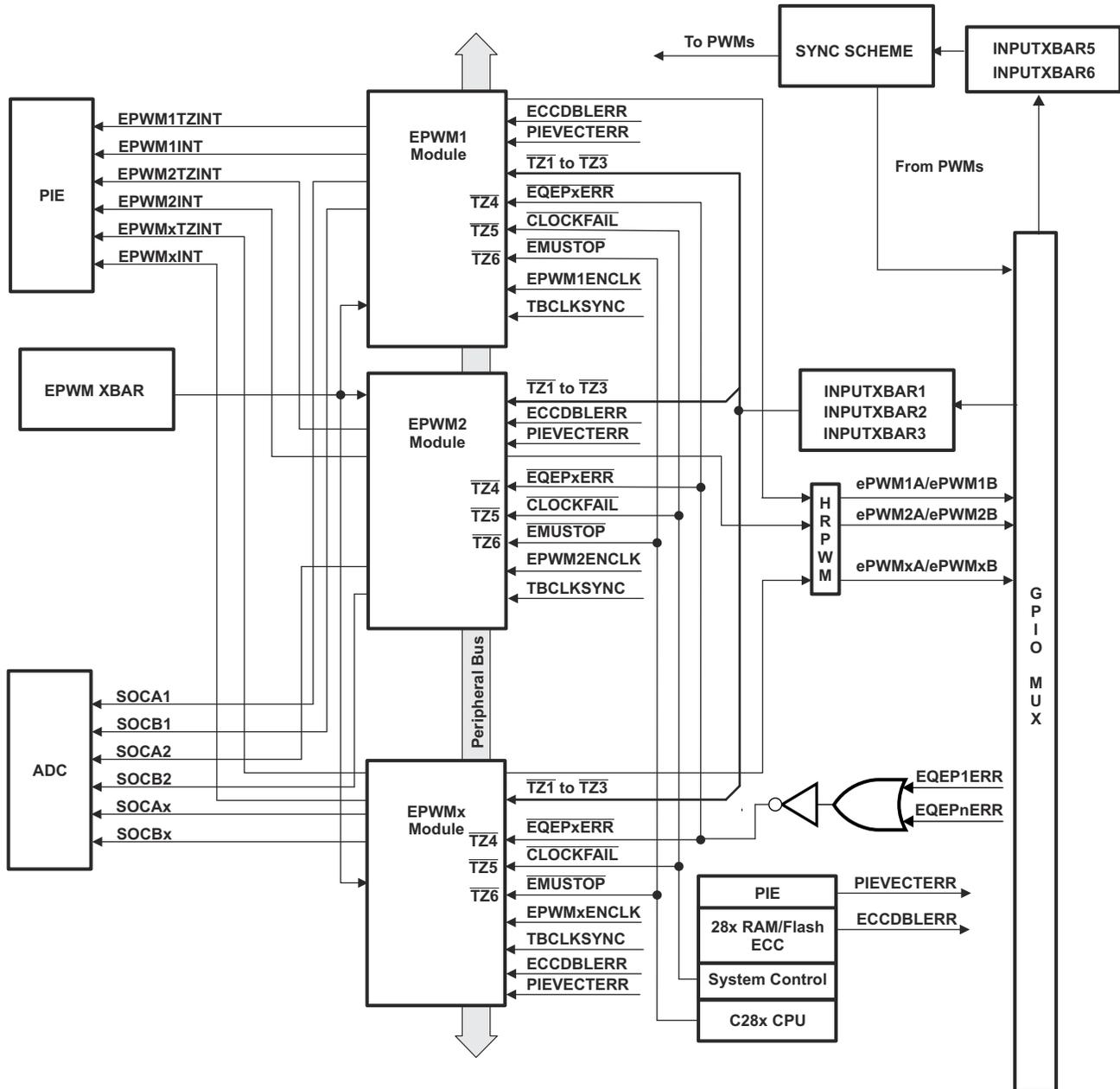
The ePWM modules are chained together by way of a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral submodules (eCAP). The number of submodules is device-dependent and based on target application needs. Submodules can also operate standalone.

Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in [Figure 13-1](#). The signals are described in detail in subsequent sections.

The order in which the ePWM modules are connected can differ from what is shown in [Figure 13-1](#). See [Section 13.4.3.3](#) for the synchronization scheme for a particular device. Each ePWM module consists of eight submodules and is connected within a system by way of the signals shown in [Figure 13-2](#).



A. This signal exists only on devices with an eQEP submodule.

Figure 13-1. Multiple ePWM Modules

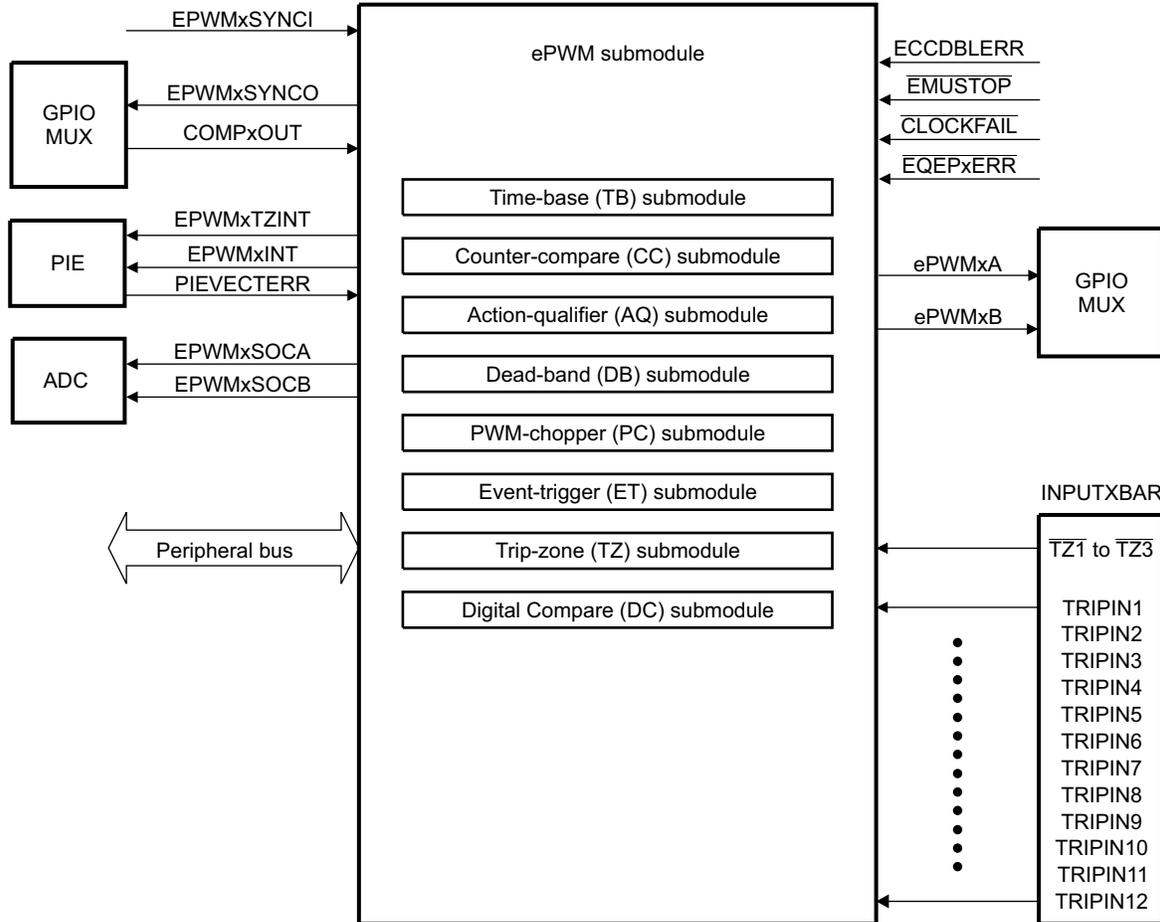


Figure 13-2. Submodules and Signal Connections for an ePWM Module

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.

- **Trip-zone signals ( $\overline{TZ1}$  to  $\overline{TZ6}$ )**

These input signals alert the ePWM module of fault conditions external to the ePWM module. Each submodule on a device can be configured to either use or ignore any of the trip-zone signals. The  $\overline{TZ1}$  to  $\overline{TZ3}$  trip-zone signals can be configured as asynchronous inputs through the GPIO peripheral using the Input X-BAR logic, refer to Figure 13-51.  $\overline{TZ4}$  is connected to an inverted EQEPx error signal (EQEPxERR), which can be generated from any one of the EQEP submodule (for those devices with an EQEP module).  $\overline{TZ5}$  is connected to the system clock fail logic, and  $\overline{TZ6}$  is connected to the EMUSTOP output from the CPU. This allows configuring a trip action when the clock fails or the CPU halts.

- **Time-base synchronization input (EPWMxSYNCl), output (EPWMxSYNCO), and peripheral (EPWMxSYNCPER) signals**

Each ePWM module can be synchronized with other ePWM modules or other peripherals, using EPWMSYNCINSEL. Each ePWM module can also generate a synchronization output signal. The source of the EPWMxSYNCO can be selected and enabled by EPWMSYNCOEN and TBCTL2.OSHTSYNCPER. For more information, see Section 13.4.3.3.

Each ePWM module also generates another PWMSYNC signal called EPWMxSYNCPER. EPWMxSYNCPER goes to the CMPSS for synchronization purposes. Functionality is configured using the HRPCTL register, but has no relation with the HRPWM. For more information on how EPWMxSYNCPER is used by the CMPSS, see the respective chapters.

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB)**

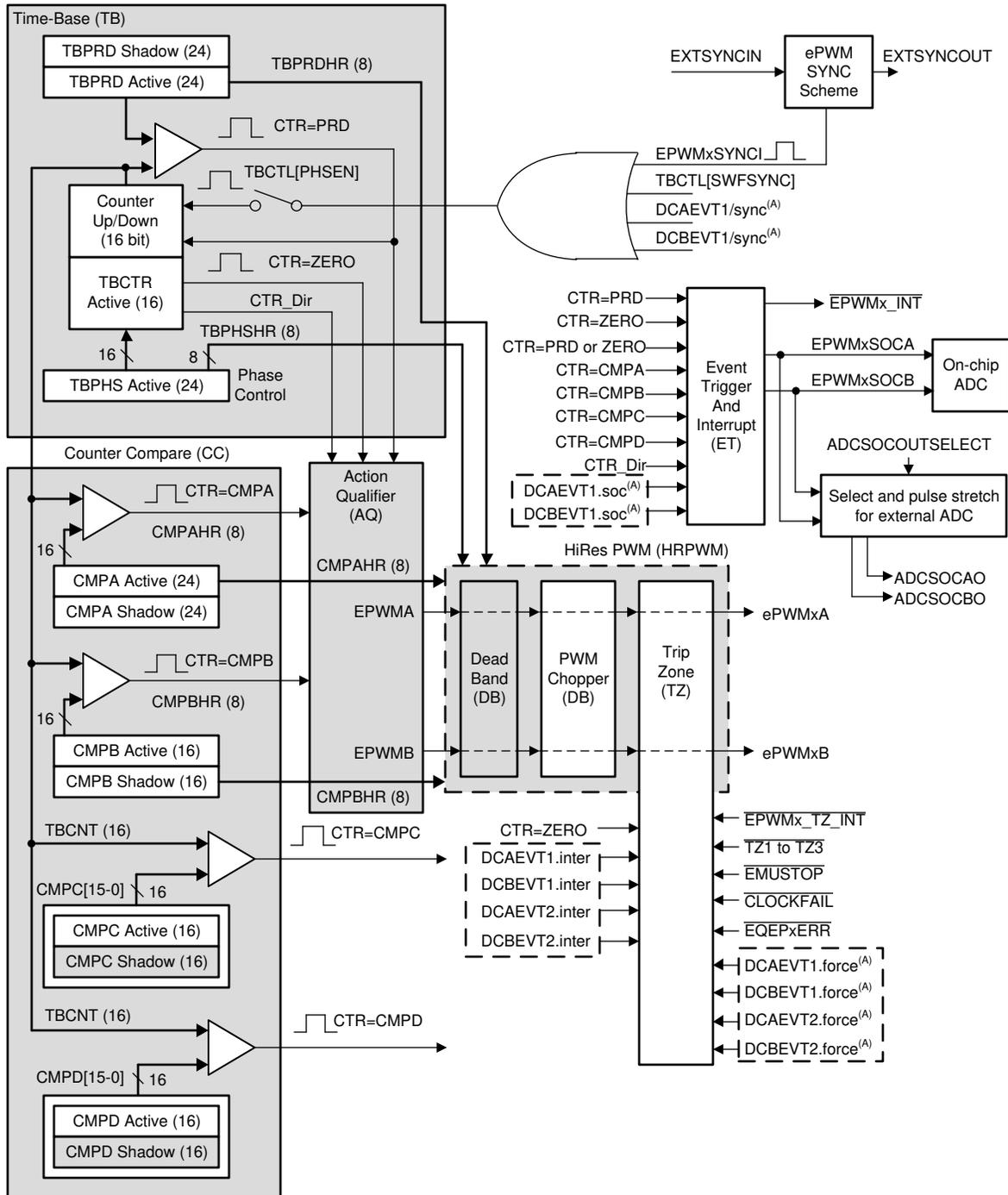
Each ePWM module has two ADC start of conversion signals. Any ePWM module can trigger a start of conversion. Whichever event triggers the start of conversion is configured in the event-trigger submodule of the ePWM.

- **Comparator output signals (COMPxOUT)**

Output signals from the comparator module can be fed through the Input X-BAR and EPWM 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.



A. These events are generated by the ePWM Digital Compare (DC) submodule based on the levels of the TRIPIN inputs.

**Figure 13-3. ePWM Modules and Critical Internal Signal Interconnects**

## 13.2 Configuring Device Pins

To connect the device input pins to the module, the Input X-BAR and EPWM X-BAR must be used. Some examples of when an external signal can be needed are TZx, TRIPx, and EXTSYNCIN. Any GPIO on the device can be configured as an input. The GPIO input qualification can 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 *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux, GPIO settings, and XBAR configuration.

## 13.3 ePWM Modules Overview

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

[Table 13-1](#) lists the key submodules together with a list of the main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, see the counter-compare submodule in [Section 13.5](#) for relevant details.

**Table 13-1. Submodule Configuration Parameters**

Submodule	Configuration Parameter or Option
<a href="#">Time Base (TB)</a>	<ul style="list-style-type: none"> <li>• Scale the time-base clock (TBCLK) relative to the ePWM clock (EPWMCLK).</li> <li>• Configure the PWM time-base counter (TBCTR) frequency or period.</li> <li>• Set the mode for the time-base counter:               <ul style="list-style-type: none"> <li>– count-up mode: used for asymmetric PWM</li> <li>– count-down mode: used for asymmetric PWM</li> <li>– count-up-and-down mode: used for symmetric PWM</li> </ul> </li> <li>• Configure the time-base phase relative to another ePWM module.</li> <li>• Synchronize the time-base counter between modules through hardware or software.</li> <li>• Configure the direction (up or down) of the time-base counter after a synchronization event.</li> <li>• Simultaneous writes to the TBPRD registers on all PWM's corresponding to the configuration on EPWMXLINK.</li> <li>• Configure how the time-base counter behaves when the device is halted by an emulator.</li> <li>• Specify the source for the synchronization output of the ePWM module</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
<a href="#">Counter Compare (CC)</a>	<ul style="list-style-type: none"> <li>• Specify the PWM duty cycle for output EPWMxA and output EPWMxB</li> <li>• Specify the time at which switching events occur on the EPWMxA or EPWMxB output</li> <li>• Specify the programmable delay for interrupt and SOC generation with additional comparators</li> <li>• Simultaneous writes to the CMPA, CMPB, CMPC, CMPD registers on all PWM's corresponding to the configuration on EPWMXLINK.</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>

**Table 13-1. Submodule Configuration Parameters (continued)**

Submodule	Configuration Parameter or Option
Action Qualifier (AQ)	<ul style="list-style-type: none"> <li>• Specify the type of action taken when a time-base counter-compare, trip-zone submodule, or comparator event occurs: <ul style="list-style-type: none"> <li>– No action taken</li> <li>– Output EPWMxA and EPWMxB switched high</li> <li>– Output EPWMxA and EPWMxB switched low</li> <li>– Output EPWMxA and EPWMxB toggled</li> </ul> </li> <li>• Force the PWM output state through software control</li> <li>• Configure and control the PWM dead band through software</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
Dead-Band Generator (DB)	<ul style="list-style-type: none"> <li>• Control of traditional complementary dead-band relationship between upper and lower switches</li> <li>• Specify the output rising-edge-delay value</li> <li>• Specify the output falling-edge delay value</li> <li>• Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.</li> <li>• Option to enable half-cycle clocking for double resolution.</li> <li>• Allow ePWMxB phase shifting with respect to the ePWMxA output.</li> <li>• Configure one shot and global load of registers in this module.</li> </ul>
PWM Chopper (PC)	<ul style="list-style-type: none"> <li>• Create a chopping (carrier) frequency.</li> <li>• Pulse width of the first pulse in the chopped pulse train.</li> <li>• Duty cycle of the second and subsequent pulses.</li> <li>• Bypass the PWM chopper module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>
Trip Zone (TZ)	<ul style="list-style-type: none"> <li>• Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events.</li> <li>• Specify the trip action taken when a fault occurs: <ul style="list-style-type: none"> <li>– Force EPWMxA and EPWMxB high</li> <li>– Force EPWMxA and EPWMxB low</li> <li>– Force EPWMxA and EPWMxB to a high-impedance state</li> <li>– Configure EPWMxA and EPWMxB to ignore any trip condition.</li> </ul> </li> <li>• Configure how often the ePWM reacts to each trip-zone signal: <ul style="list-style-type: none"> <li>– One-shot</li> <li>– Cycle-by-cycle</li> </ul> </li> <li>• Enable the trip-zone to initiate an interrupt.</li> <li>• Bypass the trip-zone module entirely.</li> <li>• Programmable option for cycle-by-cycle trip clear</li> <li>• If desired, independently configure trip actions taken when time-base counter is counting down.</li> </ul>
Event Trigger (ET)	<ul style="list-style-type: none"> <li>• Enable the ePWM events that trigger an interrupt.</li> <li>• Enable ePWM events that trigger an ADC start-of-conversion event.</li> <li>• Specify the rate at which events cause triggers (every occurrence or every 2nd or up to 15th occurrence)</li> <li>• Poll, set, or clear event flags</li> </ul>
Digital Compare (DC)	<ul style="list-style-type: none"> <li>• Enables comparator (COMP) module outputs and trip zone signals which are configured using the Input X-BAR to create events and filtered events</li> <li>• Specify event-filtering options to capture TBCTR counter, generate blanking window, or insert delay in PWM output or time-base counter based on captured value.</li> </ul>

### 13.4 Time-Base (TB) Submodule

Each ePWM module has a 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 submodule within the ePWM.

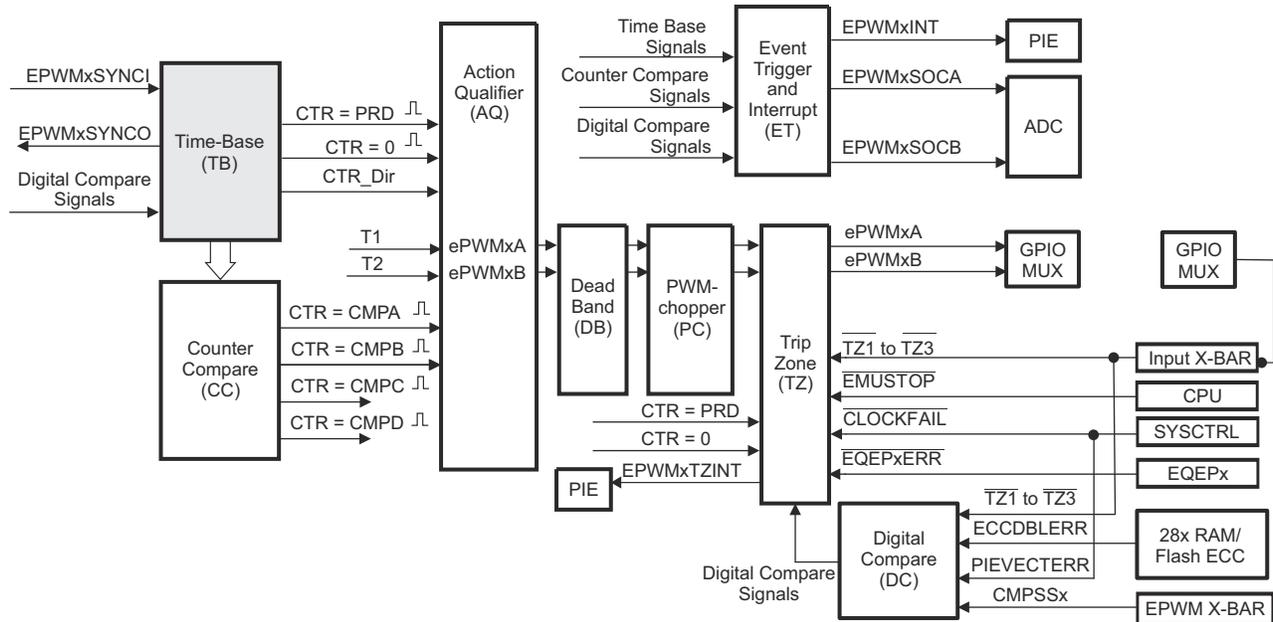


Figure 13-4. Time-Base Submodule

#### 13.4.1 Purpose of the Time-Base Submodule

The time-base submodule can be configured 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

If required by the application code to update the TBCTR value through software while the TBCTR is counting, note that the time-base module needs at least 1 TBCLK cycle for the time-base related events to be realized. Hence, the TBCTR can be written with TBCTR = PRD-1 instead of TBCTR = PRD (in case the counter is counting up) and can be written as TBCTR = 1 instead of TBCTR = 0 (in case the counter is counting down) for the events to be realized.



**Table 13-2. Key Time-Base Signals**

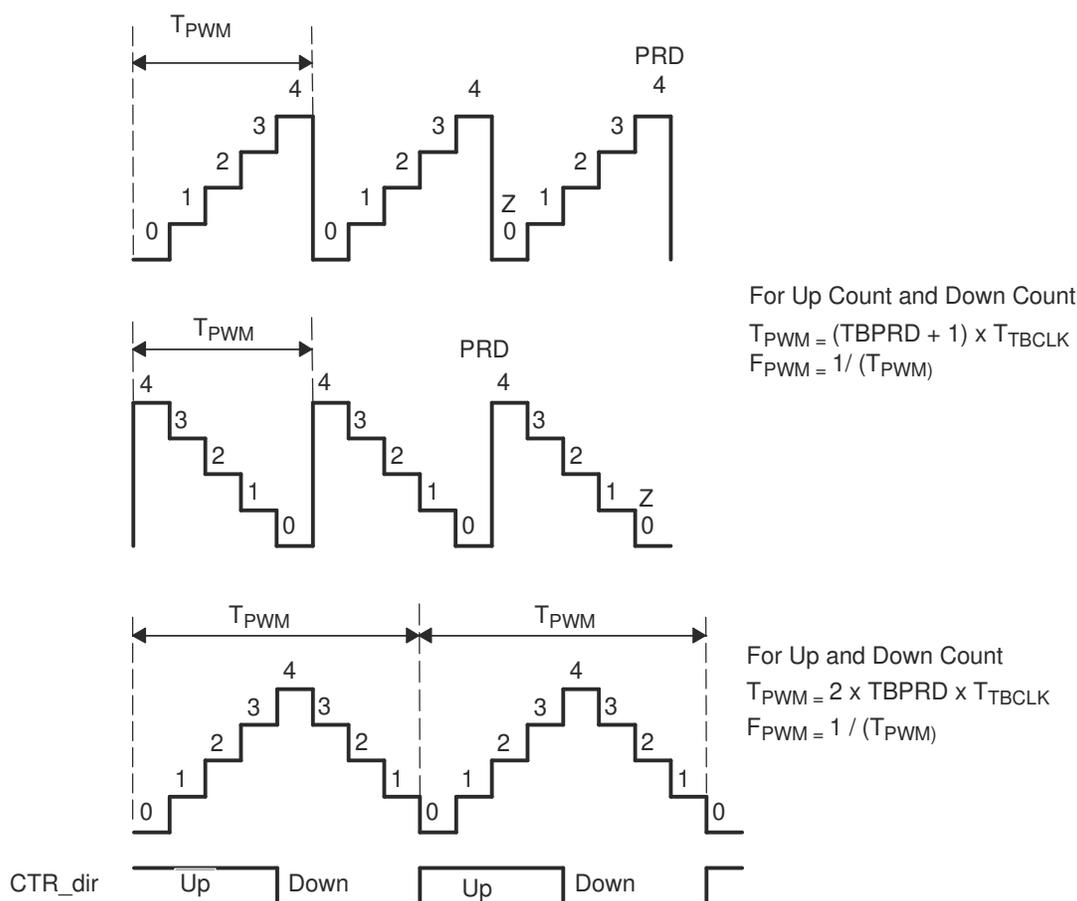
Signal	Description
EPWMxSYNCl	Time-base synchronization input.  Input pulse used to synchronize the time-base counter with the counter of other ePWM modules. For more information on all of the signals available for synchronization, see EPWMSYNClNSEL. For information on the synchronization order of a particular device, see <a href="#">Section 13.4.3.3</a> .
EPWMxSYNCO	Time-base synchronization output.  This output pulse is used to synchronize the counter of other ePWM modules. Using EPWMSYNCOUEN, TBCTL2, TBCTL3 and GLDCTL2, the source of the output pulse is selected.
EPWMxSYNCPER	Time-base peripheral synchronization output.  This output signal is used to synchronize the CMPSS to the EPWM. The output signal can be configured using the HRPCTL register. Note that this signal has no relation with the HRPWM.
CTR = PRD	Time-base counter equal to the specified period.  This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.
CTR = Zero	Time-base counter equal to zero  This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x00.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCTR = CMPB).  This event is generated by the counter-compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction.  Indicates the current direction of the ePWM's time-base counter. The signal is high when the counter is increasing and the signal is low when the counter is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF)  Generated event when the TBCTR value reaches the 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.

### 13.4.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 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 the counter reaches zero. At this point, the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In up-count mode, the time-base counter starts from zero and increments until the counter 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 the counter reaches zero. When the counter reaches zero, the time-base counter is reset to the period value and begins to decrement once again.



**Figure 13-6. Time-Base Frequency and Period**

### 13.4.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register:** The active register controls the hardware and is responsible for actions that the hardware causes or invokes.
- **Shadow Register:** The shadow register buffers provide a temporary holding location for the active register and have no direct effect on any control hardware. At a strategic point in time, the shadow register 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 13.4.3.3](#).

The global load control mechanism can also be used with the time-base period register by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in Global Shadow to Active Load Control Register (GLDCTL). Global load control mechanism is explained in [Section 13.4.7](#).

- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (TBCTL[PRDL] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

### 13.4.3.2 Time-Base Clock Synchronization

The TBCLKSYNC bit in the peripheral clock enable registers allows all users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For 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

### 13.4.3.3 Time-Base Counter Synchronization

The ePWM synchronization scheme allows for increased flexibility of synchronization of the ePWM modules. Each ePWM module has a synchronization input (SYNCl), a synchronization output (SYNCO) and a peripheral synchronization output (SYNCPER). In Figure 13-7, EXTSYNClN1 is sourced from INPUTXBAR5 and EXTSYNClN2 is sourced from INPUTXBAR6, which can be configured to select any GPIO as the synchronization input. Refer to Section 13.4.3.4 for a list of all sync inputs including INPUTXBAR5 and INPUTXBAR6. Figure 13-8 shows the sources that can be used for EXTSYNCOUOut.

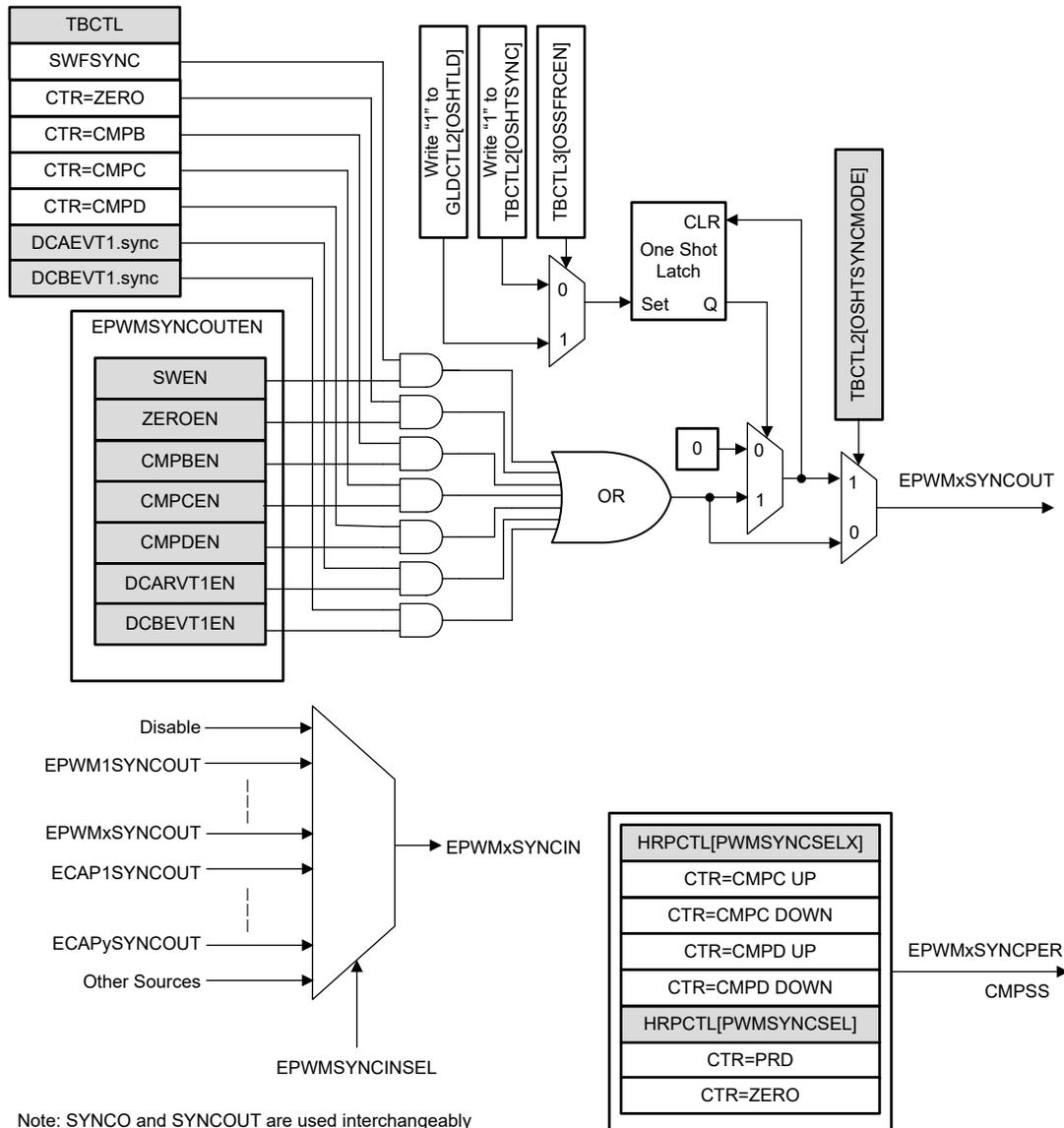
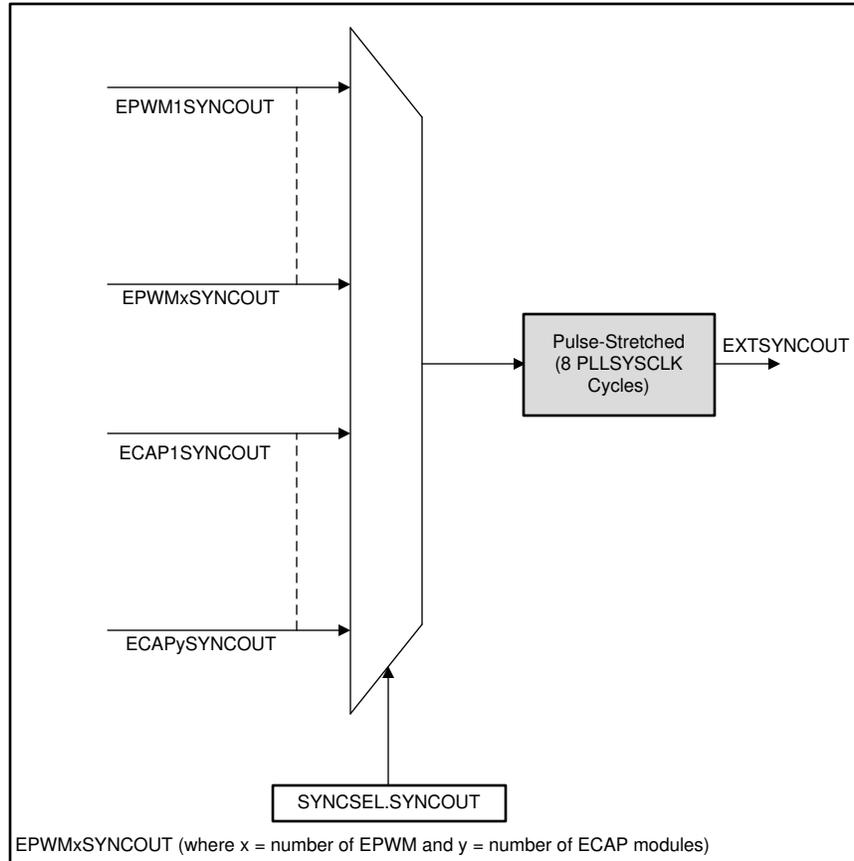


Figure 13-7. Time-Base Counter Synchronization Scheme



**Figure 13-8. ePWM External SYNC Output**

**Note**

See the data sheet 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 is 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 control module to target modules is given by:

- if (TBCLK = EPWMCLK): 2 x EPWMCLK
- if (TBCLK < EPWMCLK): 1 x TBCLK
- **Software Forced Synchronization Pulse:** Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on 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.

**Note**

If the EPWMxSYNCl signal is held high, the sync does not continuously occur. The EPWMxSYNCl is rising edge activated.

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[PHSDIR] 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 PHSDIR bit is ignored in count-up or count-down modes. See [Figure 13-9](#) through [Figure 13-12](#) for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse.

#### 13.4.3.4 ePWM SYNC Selection

[Table 13-3](#) specifies the sources for the ePWM SYNC input and output.

**Table 13-3. ePWM SYNC Selection**

EPWMSYNCINSEL.SEL, ECAPSYNCINSEL.SEL	SYNC Source
0x0	Reserved
0x1	EPWM1.SYNCOUT
0x2	EPWM2.SYNCOUT
0x3	EPWM3.SYNCOUT
0x4	EPWM4.SYNCOUT
0x5	EPWM5.SYNCOUT
0x6	EPWM6.SYNCOUT
0x7	EPWM7.SYNCOUT
0x8	Reserved
0x9	Reserved
0xA	Reserved
0xB	Reserved
0xC	Reserved
0xD	Reserved
0xE	Reserved
0xF	Reserved
0x10	Reserved
0x11	ECAP1.SYNCOUT
0x12	ECAP2.SYNCOUT
0x13	Reserved
0x14	Reserved
0x15	Reserved
0x16	Reserved
0x17	Reserved
0x18	INPUT-XBAR.INPUT5
0x19	INPUT-XBAR.INPUT6
0x1A	Reserved
0x1B	Reserved
0x1C	Reserved
0x1D	Reserved
0x1E	Reserved
0x1F	Reserved

### 13.4.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. This bit is part of the device's clock enable registers and is described in the *System Control and Interrupts* section of this manual. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is:

1. Enable the individual ePWM module clocks. This is described in the *System Control and Interrupts* chapter.
2. Set TBCLKSYNC= 0. This stops the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

### 13.4.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules

For variable frequency applications, there is a need for simultaneous writes of TBPRD and CMPx registers between ePWM modules. This prevents situations where a CTR = 0 or CTR = PRD pulse forces a shadow to active load of these registers before all registers are updated between ePWM modules (resulting in some registers being loaded from new shadow values while others are loaded from old shadow values). To support this, an ePWM register linking scheme for TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, CMPC, and CMPD registers between PWM modules has been added.

Refer to the register description for EPWMXLINK to see the linked register bit-field values for corresponding ePWM. An example of using the EPWMXLINK is linking ePWM2 CMPA with CMPA of ePWM1 through ePWM2's EPWMXLINK[CMPALINK] register bit-field. In this case, a write to CMPA of ePWM1 also changes the CMPA value for ePWM2.

### 13.4.6 Time-Base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode that is asymmetrical
- Down-count mode that is asymmetrical
- Up-down-count that 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.

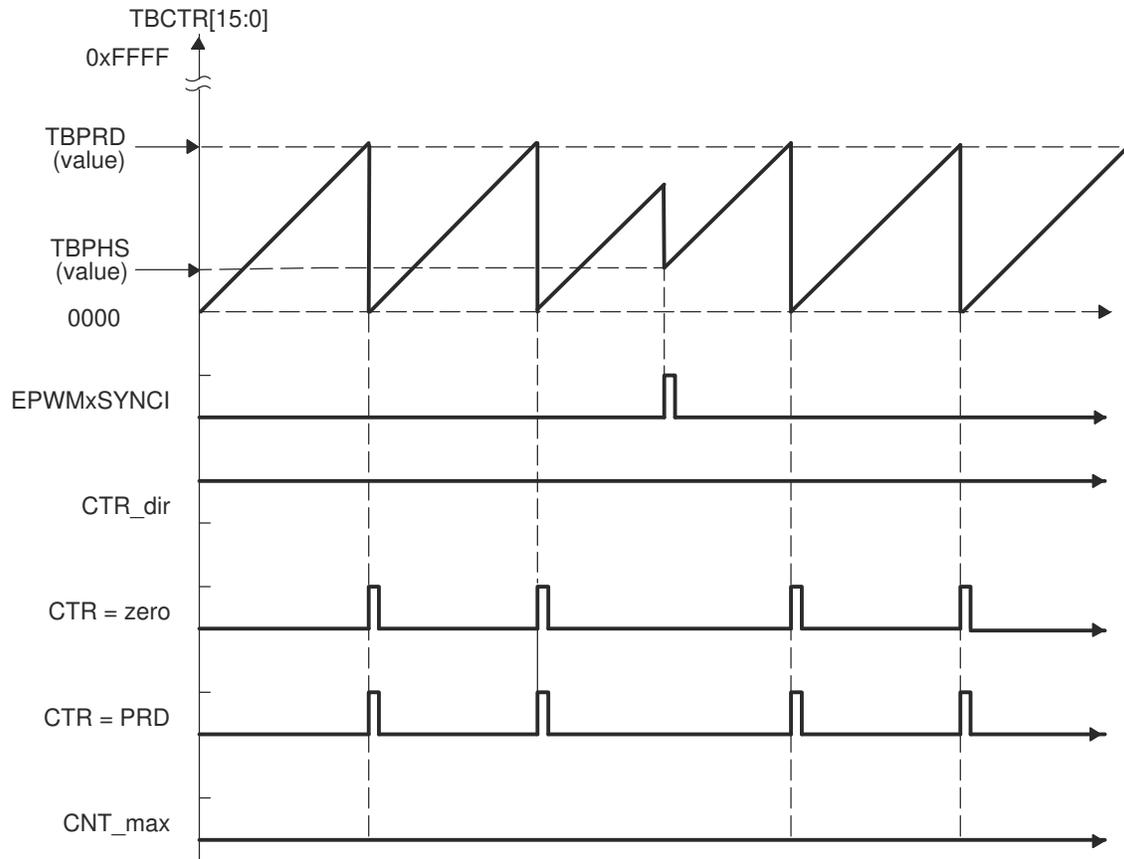


Figure 13-9. Time-Base Up-Count Mode Waveforms

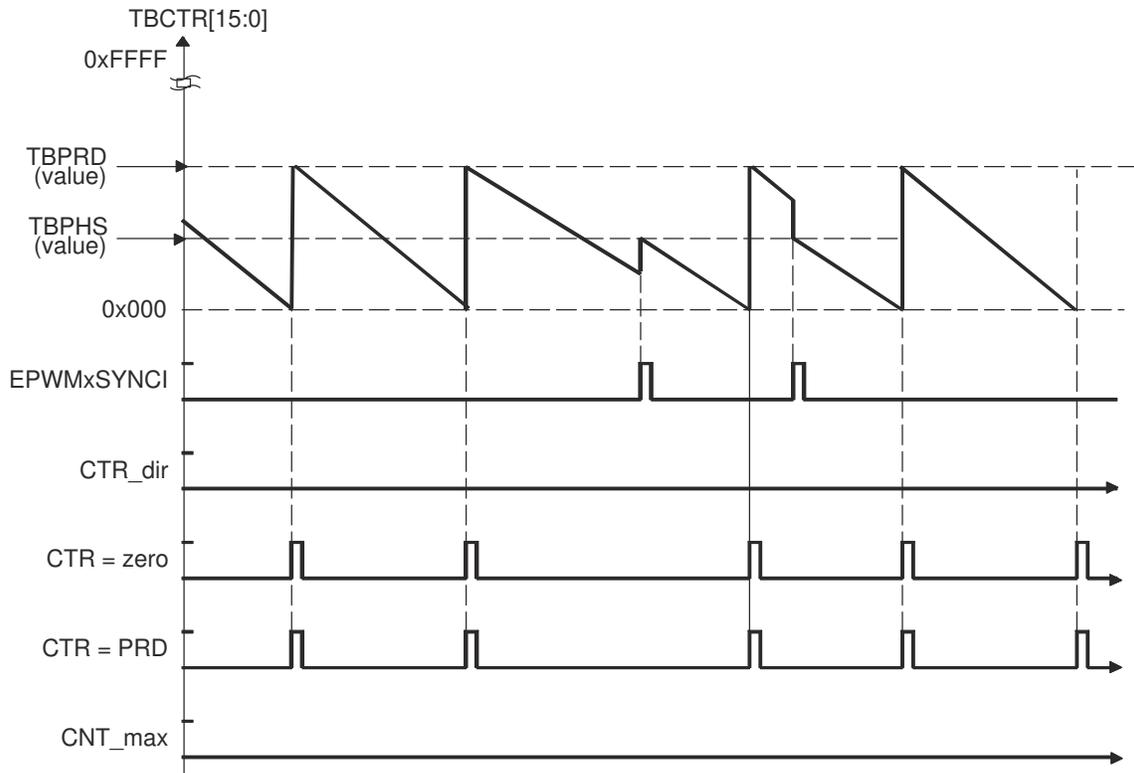
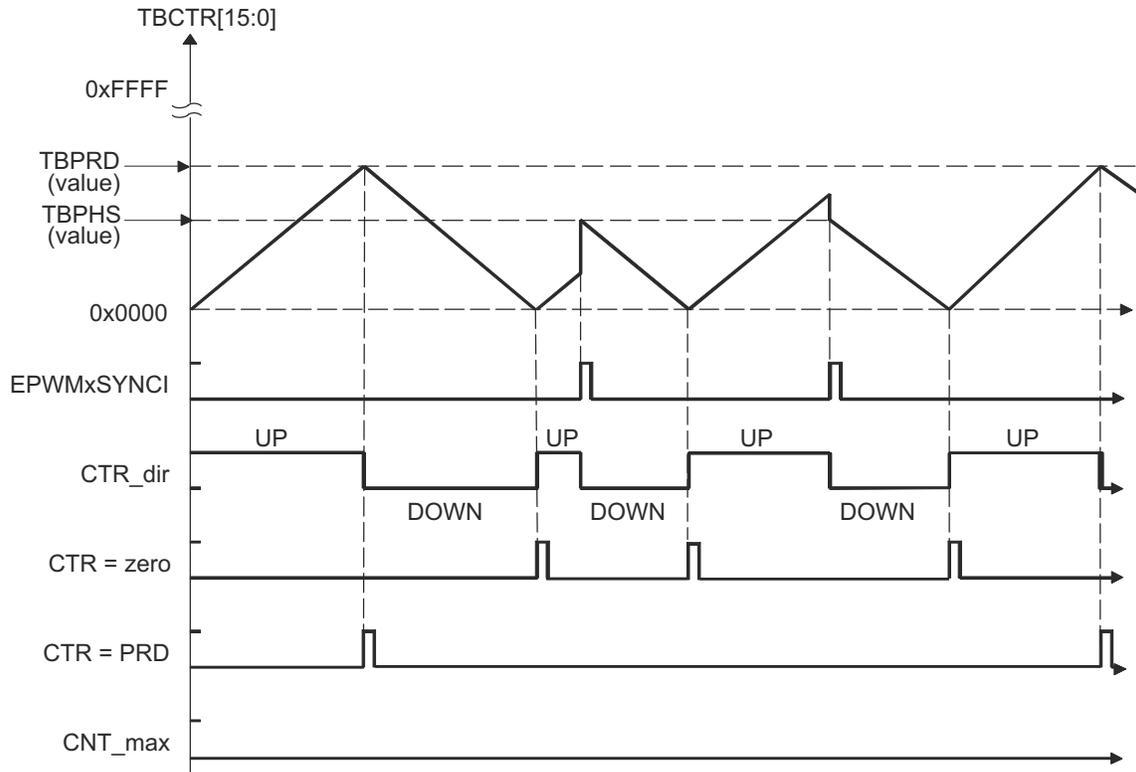


Figure 13-10. Time-Base Down-Count Mode Waveforms



**Figure 13-11. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**

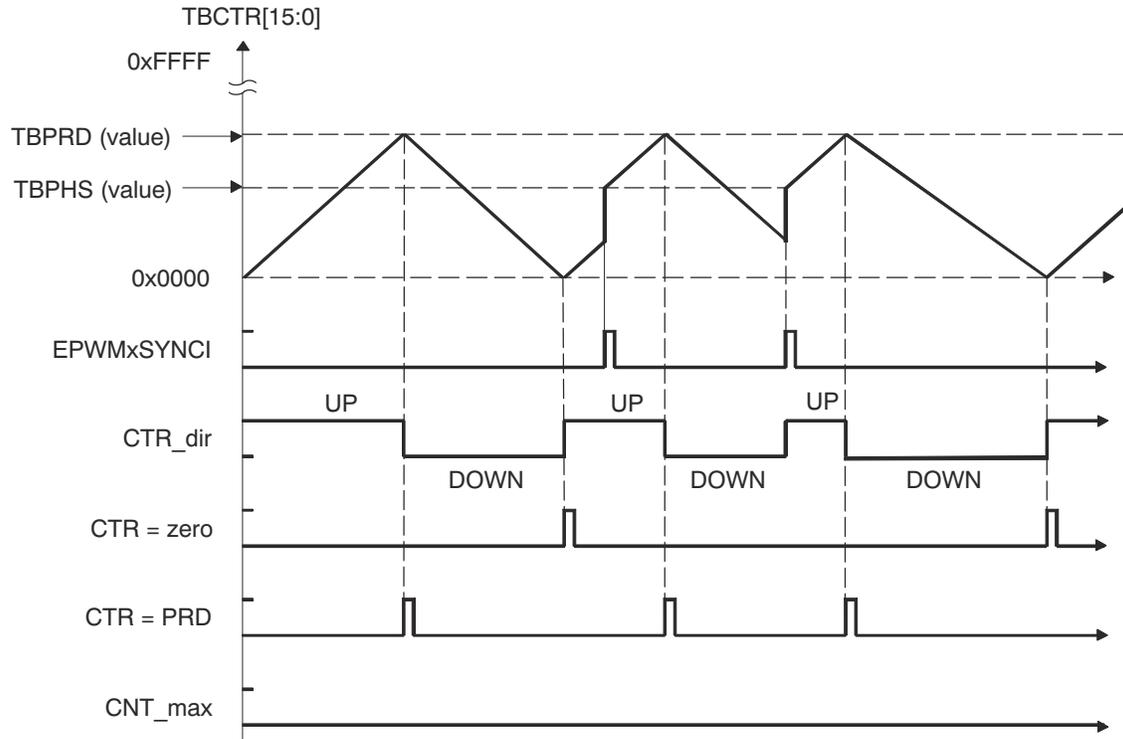


Figure 13-12. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event

### 13.4.7 Global Load

Figure 13-13 shows the signals and registers associated with the global load feature.

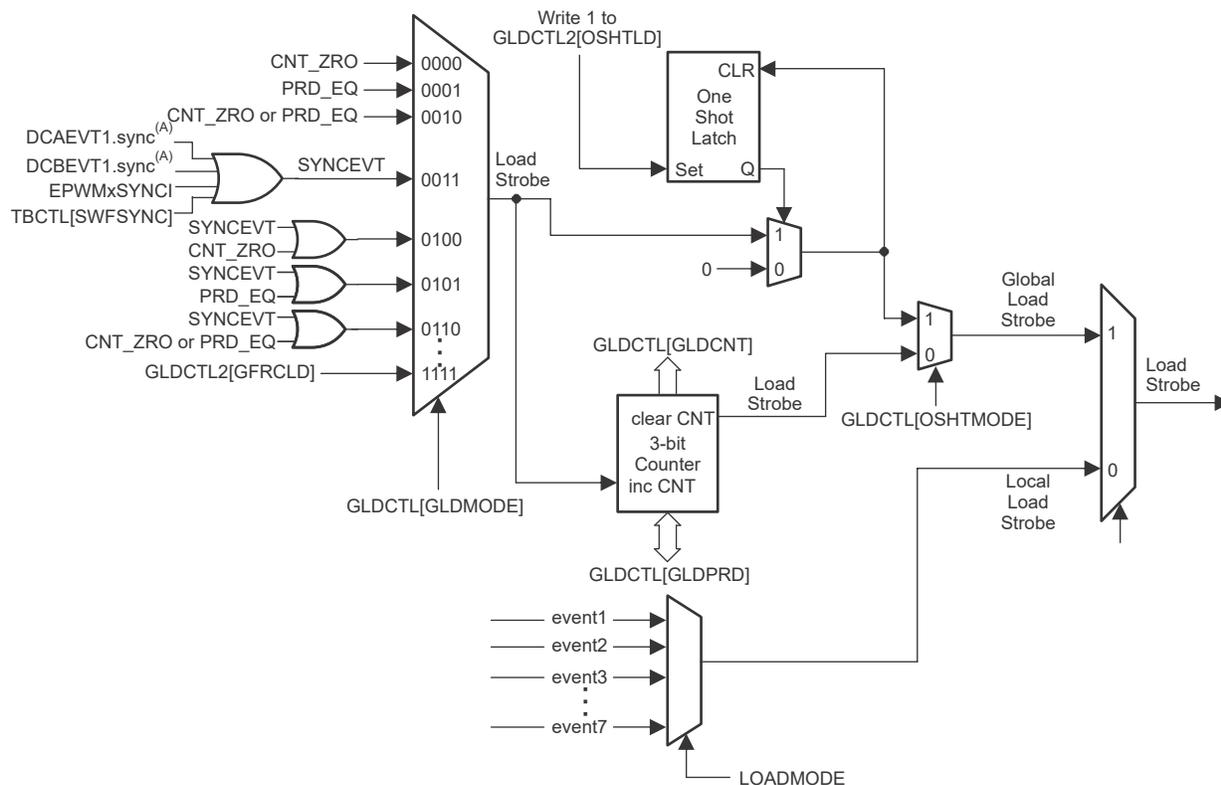


Figure 13-13. Global Load: Signals and Registers

**Note**

The SYNCEVT signal is only propagated through when PHSEN is SET.

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.

#### 13.4.7.1 Global Load Pulse Pre-Scalar

This feature provides the capability to choose shadow to active transfers to happen once in 'N' occurrences of selected global load pulse (GLDCTL[GLDMODE]). This pre-scale functionality is not available for registers that cannot or are not configured to use the global load mechanism (that is, GLDCTL[GLD] = 0 or GLDCFG[REGx] = 0).

### 13.4.7.2 One-Shot Load Mode

This feature allows users to cause the shadow register to active register transfers to occur once. When  $GLDCTL2[OSHTLD] = 1$  the shadow to active register transfer, for registers that are configured to use the global load mechanism, takes place on the event selected by  $GLDCTL[GLDMODE]$ .

Software force loading of contents from shadow register to active register is possible by using  $GLDCTL2[GFRCLD]$ . The  $GLDCTL2$  register can also be linked across multiple PWM modules by using  $EPWMXLINK[GLDCTL2LINK]$ . This, along with the one-shot load mode feature discussed above, provides a method to correctly update multiple PWM registers in one or more PWM modules at certain PWM events or, if desired, in the same clock cycle. This is very useful in variable frequency applications and/or multi-phase interleaved applications.

#### Note

One-shot load mode must not be used when high-resolution mode is enabled.

### 13.4.7.3 One-Shot Sync Mode

To enable the one-shot sync mode to generate a SYNCOUT pulse, configure the  $TBCTL2[OSHTSYNCMODE]$  bit and set the  $TBCTL2[OSHTSYNC]$  bit as shown in Figure 13-14.

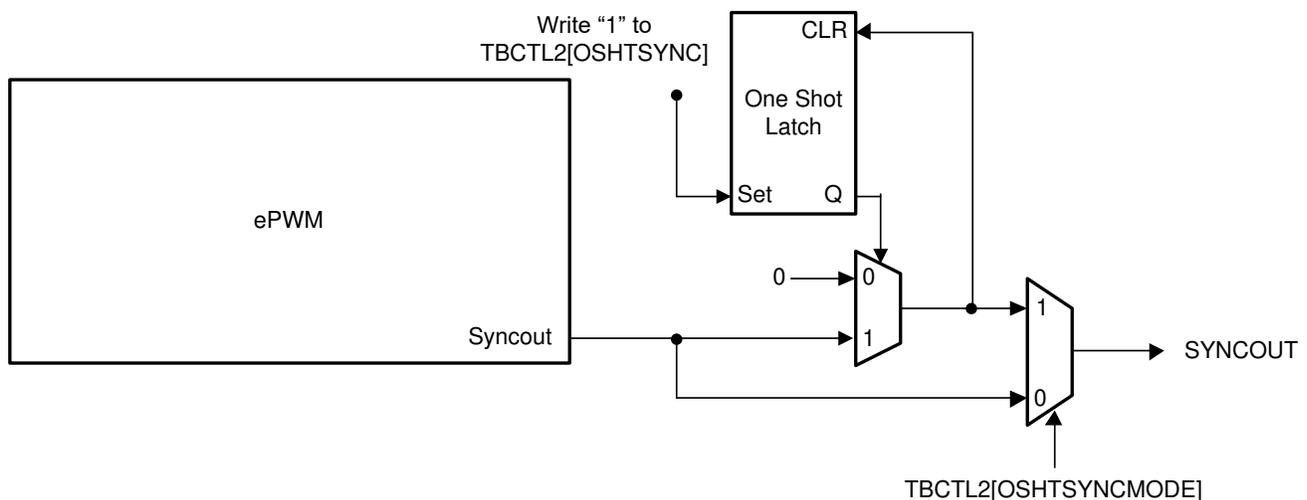
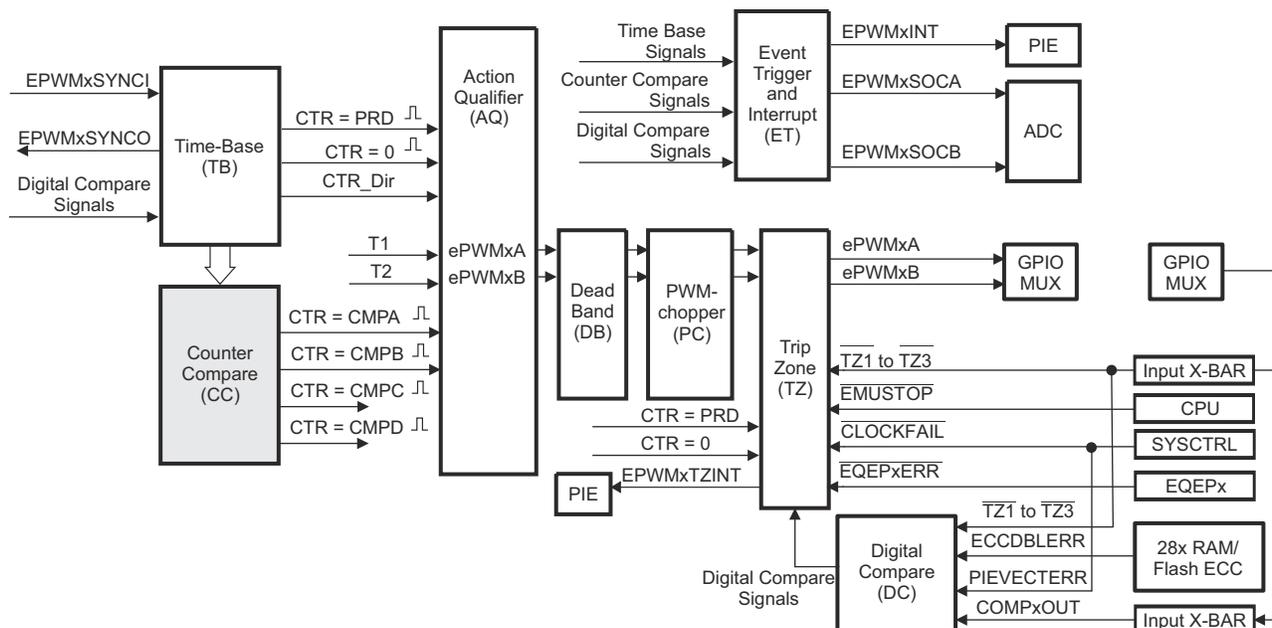


Figure 13-14. One-Shot Sync Mode

## 13.5 Counter-Compare (CC) Submodule

Figure 13-15 illustrates the counter-compare submodule within the ePWM.



**Figure 13-15. Counter-Compare Submodule**

### 13.5.1 Purpose of the Counter-Compare Submodule

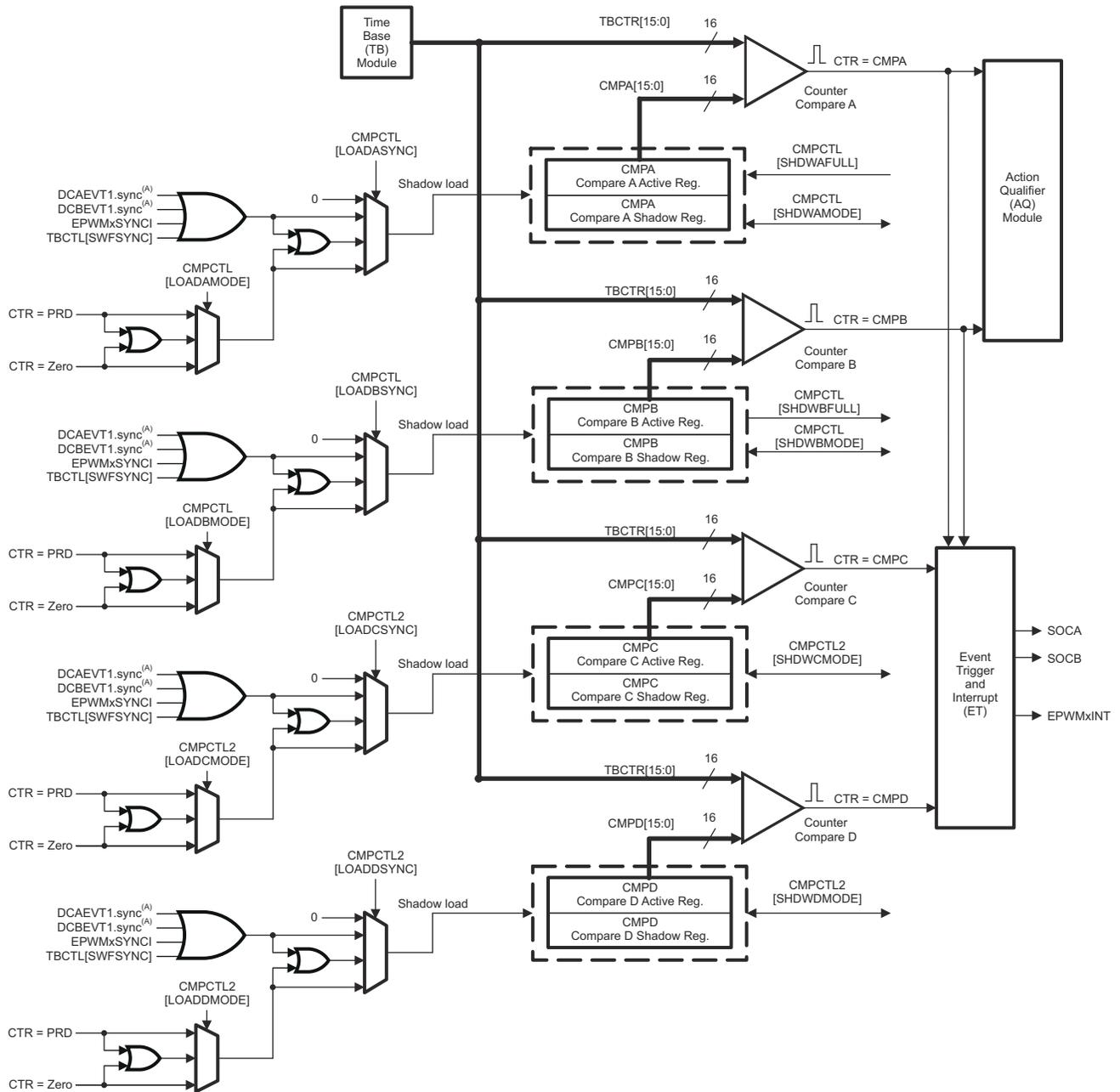
The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA), counter-compare B (CMPB), counter-compare C (CMPC), and counter-compare D (CMPD) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

The counter-compare:

- Generates events based on programmable time stamps using the CMPA, CMPB, CMPC, and CMPD registers:
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
  - CTR = CMPC: Time-base counter equals counter-compare C register (TBCTR = CMPC)
  - CTR = CMPD: Time-base counter equals counter-compare D register (TBCTR = CMPD)
- Controls the PWM duty cycle, if the action-qualifier submodule is configured appropriately using counter-compare A (CMPA) and counter-compare B (CMPB)
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

### 13.5.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is shown in Figure 13-16.



A. These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs (for example, CMPSSx and TZ signals).

**Figure 13-16. Detailed View of the Counter-Compare Submodule**

### 13.5.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating events that can be used in the action-qualifier and event-trigger submodules. There are four independent compare events:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).
3. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC). This event can be used to generate an event in the event trigger submodule only.
4. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD). This event can be used to generate an event in the event trigger submodule only.

For up-count or down-count mode, each event occurs only once per cycle. For up-down count mode, each event occurs twice per cycle if the compare value is between 0x00-TBPRD; and once per cycle if the compare value is equal to 0x00 or equal to TBPRD. These events are applied to the action-qualifier submodule where the events are qualified by the counter direction and converted into actions if enabled. Refer to [Section 13.6.1](#) for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occur at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. The register that is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPC shadow register and CMPD shadow register, respectively. The behavior of the two load modes is:

#### 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], and 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.

---

#### Note

Refer to [Section 13.6.5](#) for valid configurations of CMPA/CMPB and LOADAMODE/LOADBMODE.

---

#### Immediate Load Mode:

If the immediate load mode is selected (that is, CMPCTL[SHDWAMODE] = 1 or CMPCTL[SHDWBMODE] = 1), then a read from or a write to the register goes 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 CMPC shadow register and CMPD 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], and 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 goes directly to the active register.

### Global Load Support

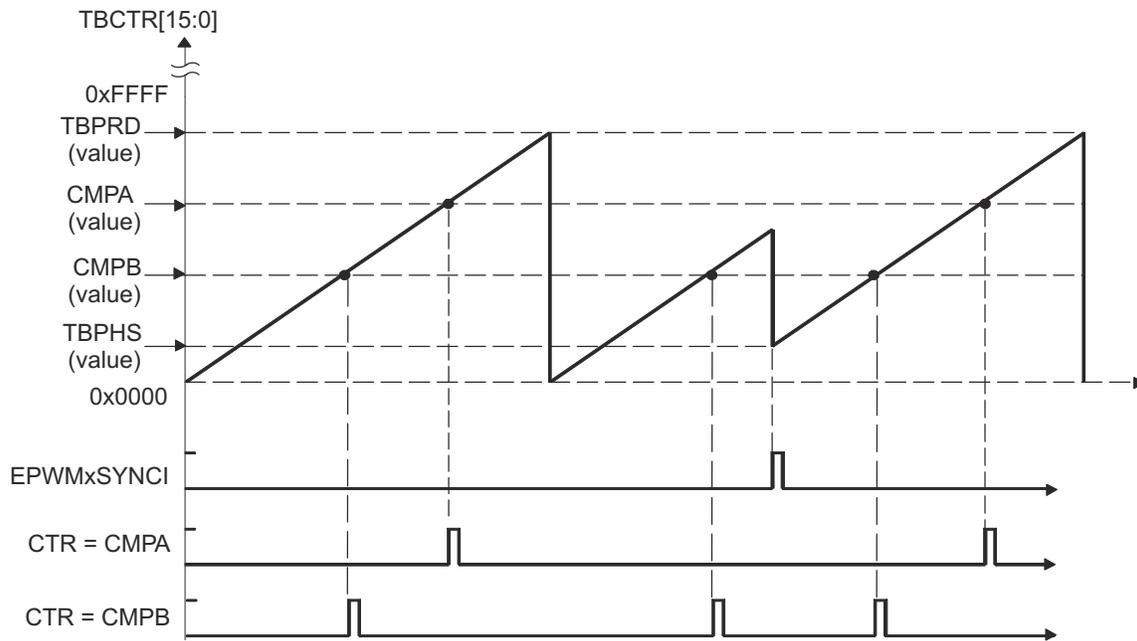
The global load control mechanism can also be used for all counter-compare registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When the global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global load control mechanism is explained in [Section 13.4.7](#).

### 13.5.4 Count Mode Timing Waveforms

The counter-compare module can generate compare events in all three count modes:

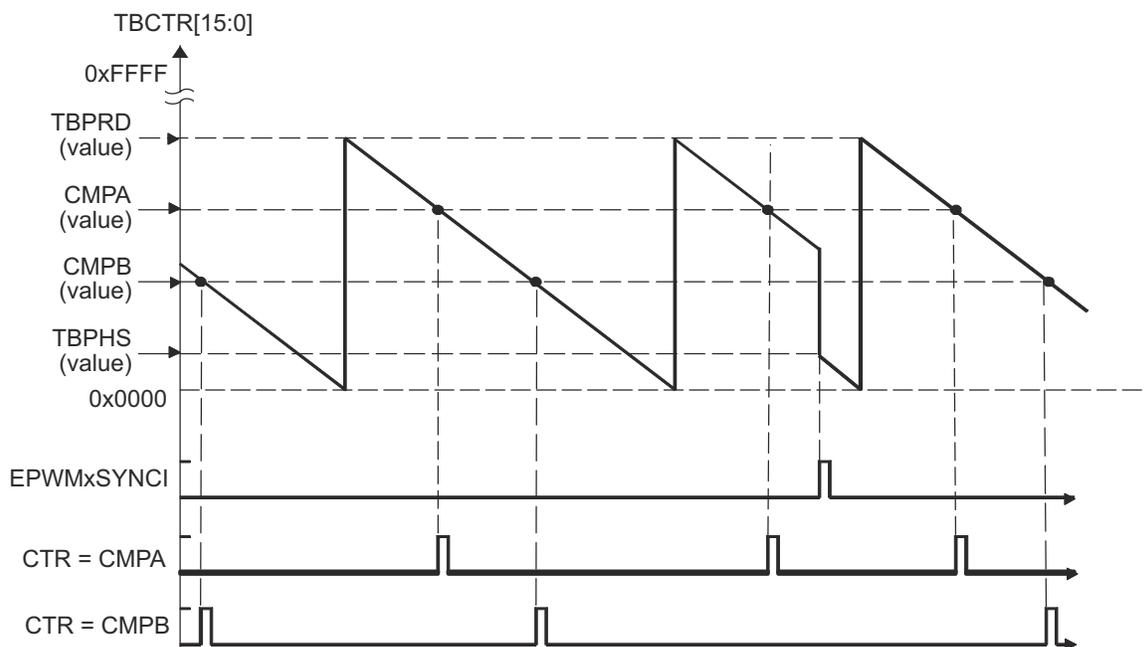
- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

To best illustrate the operation of the first three modes, the timing diagrams in [Figure 13-17](#) through [Figure 13-20](#) show when events are generated and how the EPWMxSYNCl signal interacts.



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 13-17. Counter-Compare Event Waveforms in Up-Count Mode**



**Figure 13-18. Counter-Compare Events in Down-Count Mode**

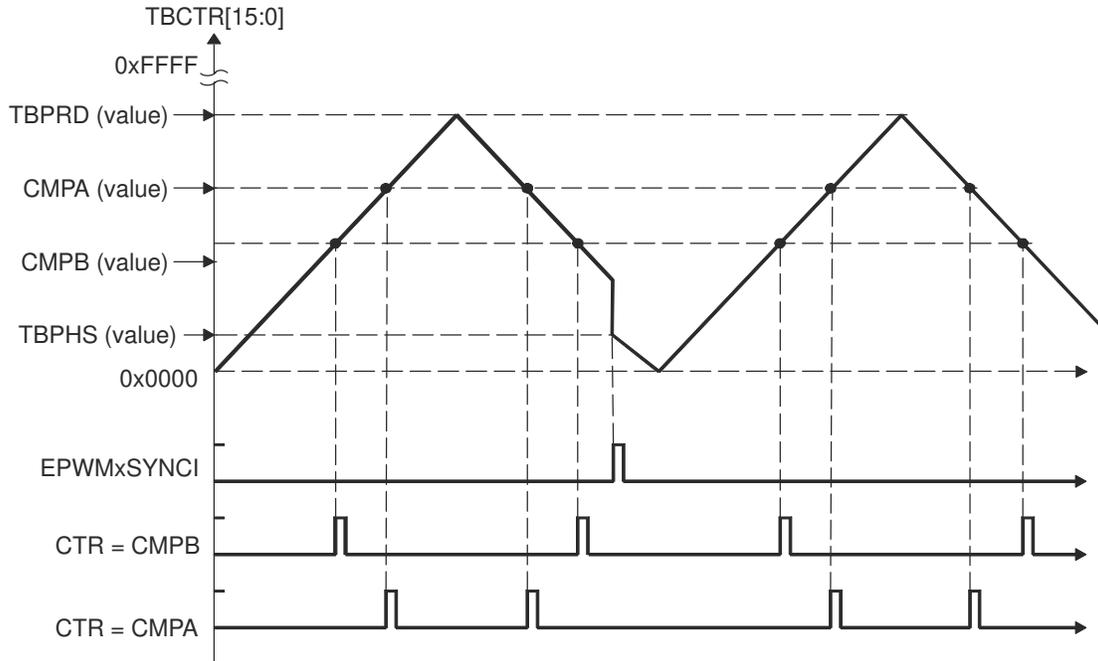


Figure 13-19. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

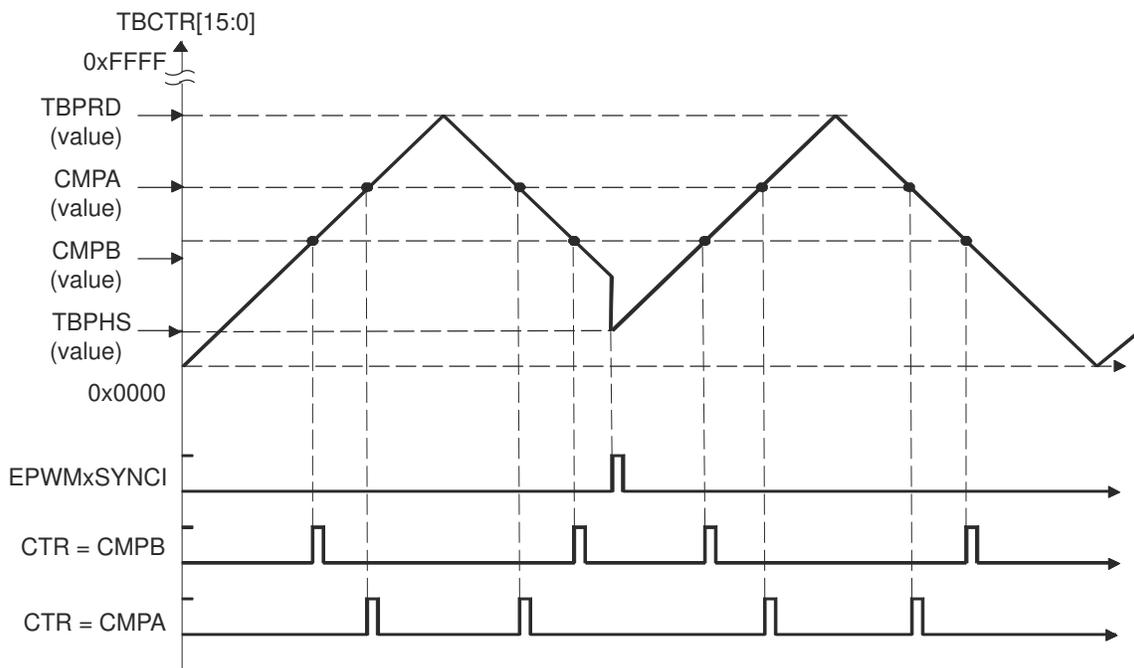
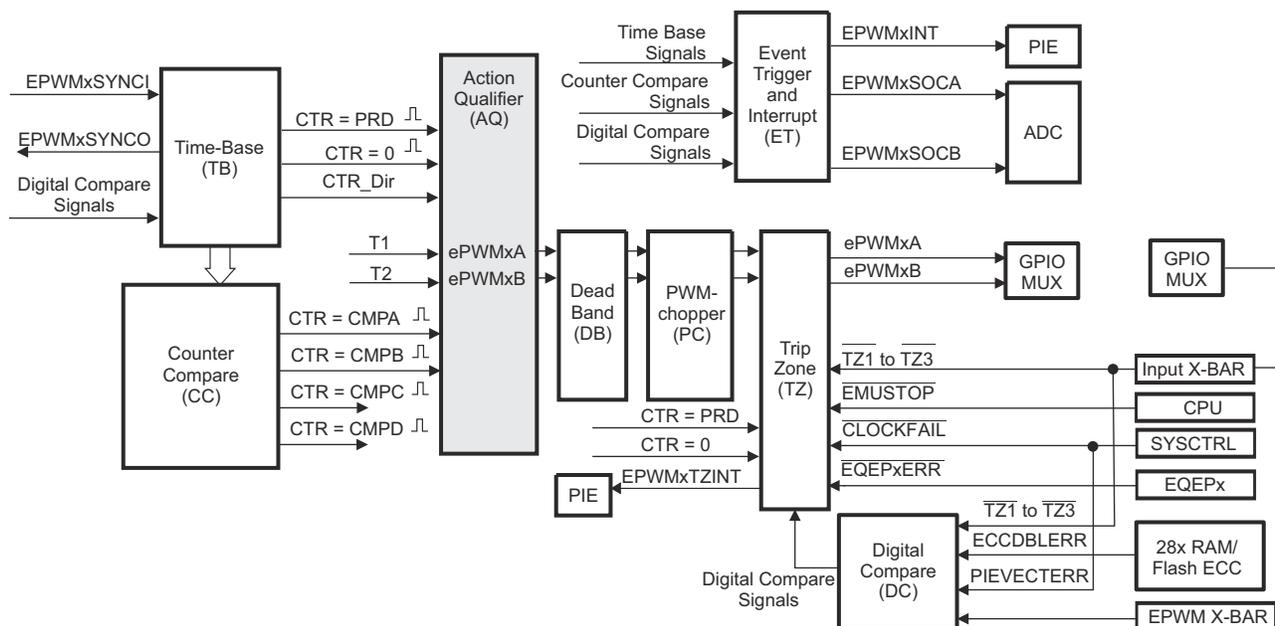


Figure 13-20. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event

## 13.6 Action-Qualifier (AQ) Submodule

The action-qualifier submodule has the most important role in waveform construction and PWM generation. The action-qualifier submodule decides which events are converted into various action types, thereby, producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

Figure 13-21 illustrates the action-qualifier submodule within the ePWM.



**Figure 13-21. Action-Qualifier Submodule**

### 13.6.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- T1, T2 events: Trigger events based on comparator, trip or syncin events
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

### 13.6.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is shown in Figure 13-22 and monitored by way of the registers in Section 13.17.

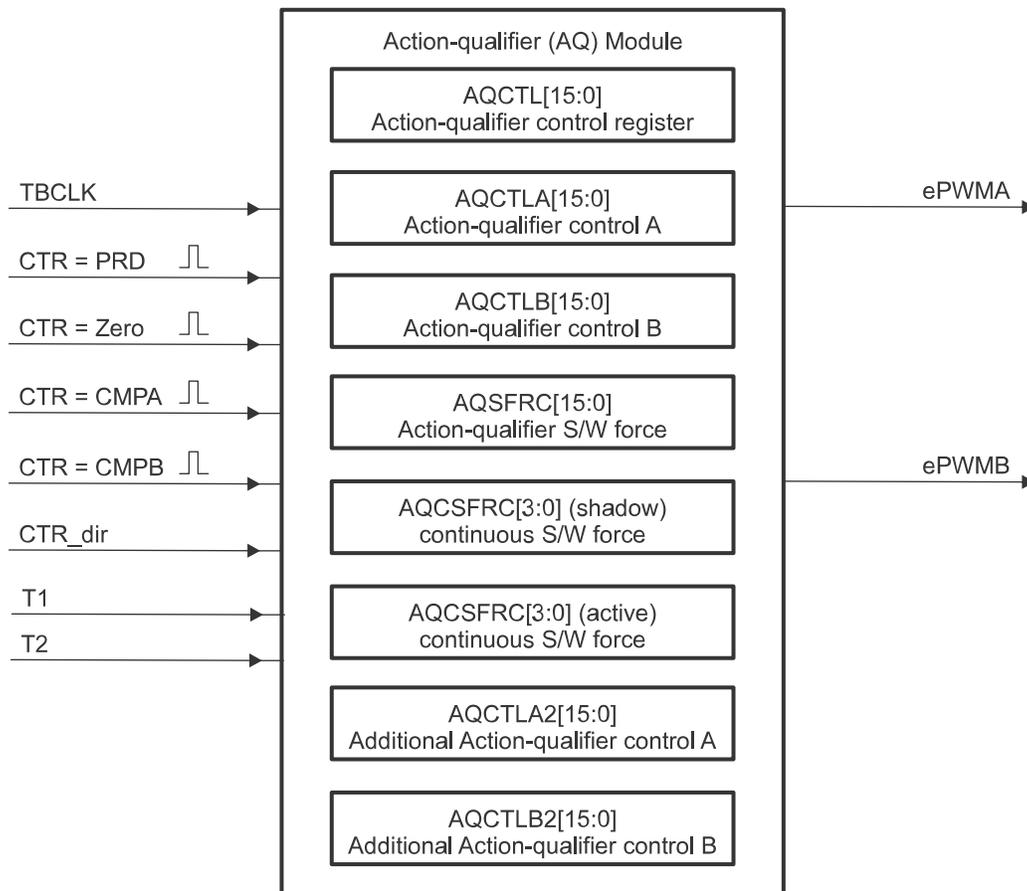


Figure 13-22. Action-Qualifier Submodule Inputs and Outputs

For convenience, the possible input events are summarized again in Table 13-4.

Table 13-4. 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 0	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.

**Note**

If the CSFA is not used in shadow mode, the RLDCSF bit must be configured to disable shadow mode.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the description in [Section 13.10](#) for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured using the control registers found in the *ePWM Registers* section.

For clarity, the illustrations in this chapter use a set of symbolic actions. These symbols are summarized in [Figure 13-23](#). 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 the time positions are programmed by way of the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"(the default at reset).

SW force	TB Counter equals			Trigger Events			Actions
	Zero	Comp A	Comp B	Period	T1	T2	
							Do Nothing
							Clear Lo
							Set Hi
							Toggle

**Figure 13-23. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

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 can or cannot

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.

### 13.6.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case, events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down count mode are shown in [Table 13-5](#). 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 13-5. 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 (Lowest)	Counter equals zero	Counter equals period (TBPRD)

[Table 13-6](#) shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up; therefore, down-count events never are taken.

**Table 13-6. 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 13-7](#) shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down; therefore, up-count events never are taken.

**Table 13-7. Action-Qualifier Event Priority for Down-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	T1 on down-count (T1D)
4	T2 on down-count (T2D)
5	Counter equal to CMPB on down-count (CBD)
6	Counter equal to CMPA on down-count (CAD)
7 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case, the action takes place as shown in [Table 13-8](#).

**Table 13-8. Behavior if CMPA/CMPB is Greater than the Period**

Counter Mode	Compare on Up-Count Event CAD/CBD	Compare on Down-Count Event CAD/CBD
Up-Count Mode	If $CMPA/CMPB \leq TBPRD$ period, then the event occurs on a compare match ( $TBCTR=CMPA$ or $CMPB$ ). If $CMPA/CMPB > TBPRD$ , then the event does not occur.	Never occurs.
Down-Count Mode	Never occurs.	If $CMPA/CMPB < TBPRD$ , the event occurs on a compare match ( $TBCTR=CMPA$ or $CMPB$ ). If $CMPA/CMPB \geq TBPRD$ , the event occurs on a period match ( $TBCTR=TBPRD$ ).
Up-Down Count Mode	If $CMPA/CMPB < TBPRD$ and the counter is incrementing, the event occurs on a compare match ( $TBCTR=CMPA$ or $CMPB$ ). If $CMPA/CMPB \geq TBPRD$ , the event occurs on a period match ( $TBCTR = TBPRD$ ).	If $CMPA/CMPB < TBPRD$ and the counter is decrementing, the event occurs on a compare match ( $TBCTR=CMPA$ or $CMPB$ ). If $CMPA/CMPB \geq TBPRD$ , the event occurs on a period match ( $TBCTR=TBPRD$ ).

### 13.6.4 AQCTLA and AQCTLB Shadow Mode Operations

To enable Action Qualifier mode changes which must occur at the end of a period even when the phase changes, shadowing of the AQCTLA and AQCTLB registers has been added on ePWMs type 2 and later. Additionally, shadow to active load on SYNC of these registers is supported as well. Shadowing of this register is enabled and disabled by the AQCTL[SHDWAQAMODE] and AQCTL[SHDWAQBMODE] bits. These bits enable and disable the AQCTLA shadow register and AQCTLB shadow register, respectively. The behavior of the two load modes is:

#### Shadow Mode:

The shadow mode for the AQCTLA is enabled by setting the AQCTL[SHDWAQAMODE] bit, and the shadow register for AQCTLB is enabled by setting the AQCTL[SHDWAQBMODE] bit. Shadow mode is disabled by default for both AQCTLA and AQCTLB

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the AQCTL[LDAQAMODE], AQCTL[LDAQBMODE], AQCTL[LDAQASYNC], and AQCTL[LDAQBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period ( $TBCTR = TBPRD$ ).
- CTR = Zero: Time-base counter equal to zero ( $TBCTR = 0x00$ )
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCl or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LDAQAMODE/LDAQBMODE

#### Global Load Support

Global load control mechanism can also be used for AQCTLA:AQCTLA2, AQCTLB:AQCTLB2, and AQCSFRC registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected, the transfer of contents from shadow register to active register for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global load control mechanism is explained in [Section 13.4.7](#).

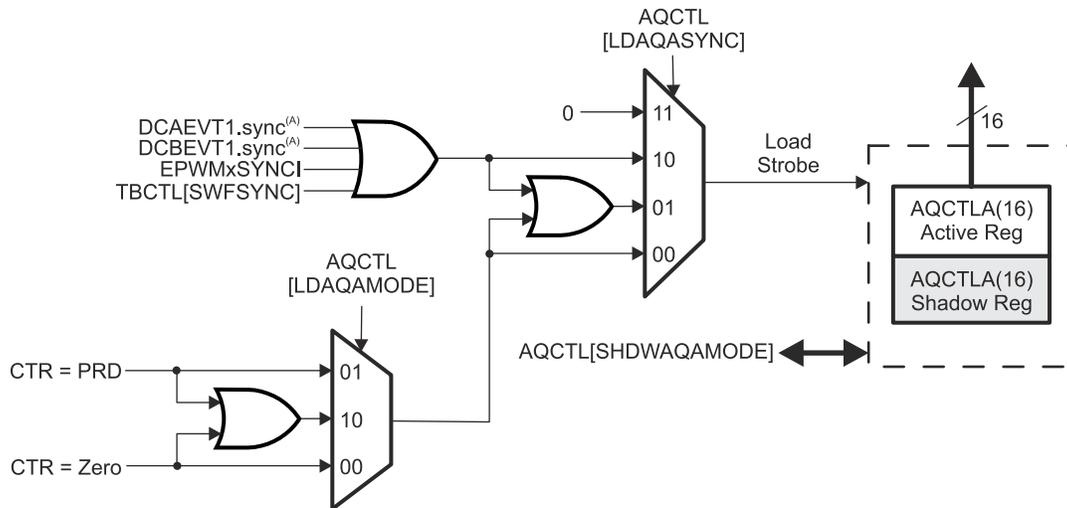
#### Immediate Load Mode:

If the immediate load mode is selected (that is, AQCTL[SHDWAQAMODE] = 0 or AQCTL[SHDWAQBMODE] = 0), then a read from or a write to the register goes directly to the active register. See [Figure 13-24](#) and [Figure 13-25](#).

**Note**

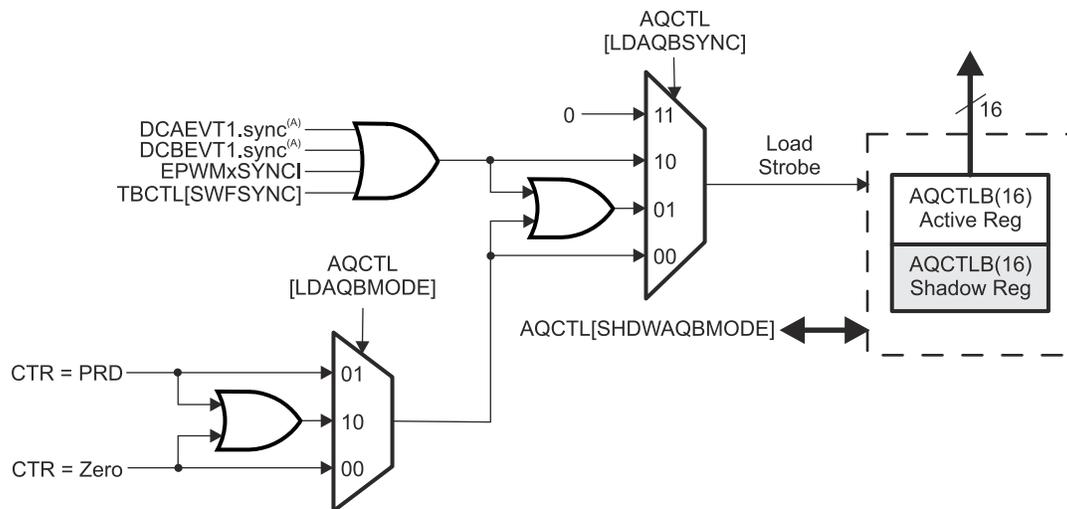
Shadow to Active Load of Action Qualifier Output A/B Control Register [AQCTLA and AQCTLB] on CMPA = 0 or CMPB = 0 boundary

If the Counter-Compare A Register (CMPA) or Counter-Compare B Register (CMPB) is set to a value of 0 and the action qualifier action on AQCTLA and AQCTLB is configured to occur in the same instant as a shadow to active load (that is, CMPA=0 and AQCTLA shadow to active load on TBCTR=0 using AQCTL register LDAQAMODE and LDAQAMODE bits), then both events enter contention. 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.



- A. These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs (for example, CMPSSx and T<sub>Z</sub> signals).

**Figure 13-24. AQCTL[SHDWAQAMODE]**



- A. These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs (for example, CMPSSx and T<sub>Z</sub> signals).

**Figure 13-25. AQCTL[SHDWAQBMODE]**

### 13.6.5 Configuration Requirements for Common Waveforms

#### Note

The waveforms in this chapter 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 the respective shadow registers once every period. Specify when the update takes place: either when the time-base counter reaches zero or when the time-base counter reaches the 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 loading CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If loading CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1.

This means there is always 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, you **must** load CMPA/CMPB on TBPRD. When CMPA/CMPB is not loaded on TBCTR=PRD, boundary conditions can occur depending on the timing of the write and the value written to CMPA/CMPB. 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.

#### When using up-count mode to generate an asymmetric PWM with deadband enabled:

- To achieve 0%-100% PWM use the following configuration: When the CMPA value is too close to 0 or PRD such that the following conditions are met ( $CMPX < \text{Deadband}$ ) or ( $CMPX > \text{PRD} - \text{Deadband}$ ), the actions specified by the AQCTL register for CMPX do not take effect. To avoid this, the AQCTL settings must be altered under these conditions only to generate either high or low pulses for both CAU or CAD events (both set or both clear). Make sure that this software update is occurring synchronous to the PWM carrier cycle, and shadow mode is enabled.

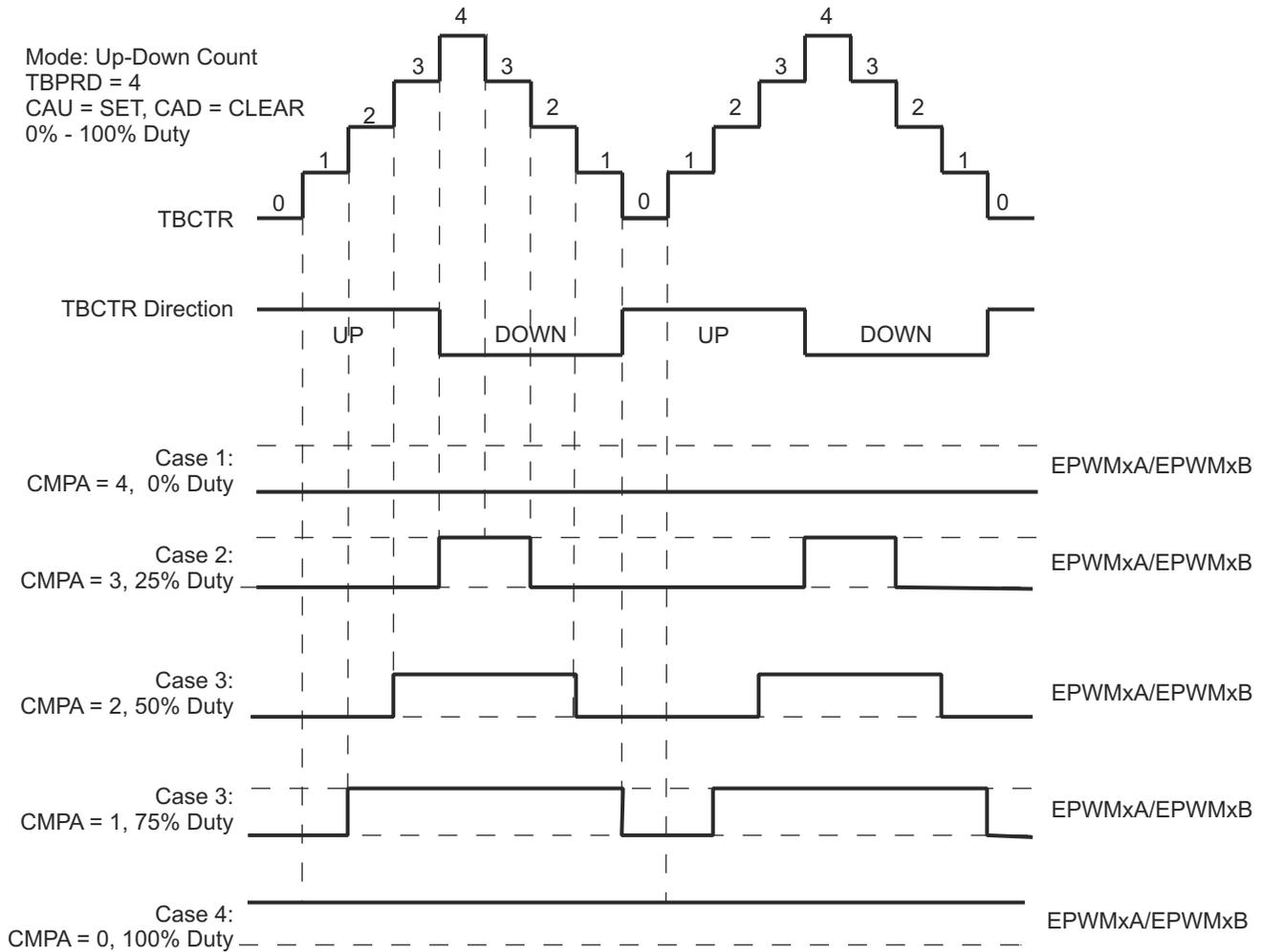
#### When using up-down count mode to generate an asymmetric PWM with deadband enabled:

- To achieve 0%-100% PWM use the following configuration: When the CMPA value is too close to 0 or PRD such that the following conditions are met ( $CMPX < \text{Deadband}/2$ ) or ( $CMPX > \text{PRD} - (\text{Deadband})/2$ ), the actions specified by the AQCTL register for CMPX do not take effect. To avoid this, the AQCTL settings must be altered under these conditions only to generate either high or low pulses for both CAU or CAD events (both set or both clear). Make sure that this software update is occurring synchronous to the PWM carrier cycle, and shadow mode is enabled.

See [Using Enhanced Pulse Width Modulator \(ePWM\) Module for 0-100% Duty Cycle Control](#).

Figure 13-26 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 pulls the PWM output high. Likewise when the counter is decrementing, the compare match pulls the PWM signal low. When  $CMPA = 0$ , the PWM signal is high for the entire period giving a 100% duty waveform. When  $CMPA = TBPRD$ , the PWM signal is low achieving 0% duty.

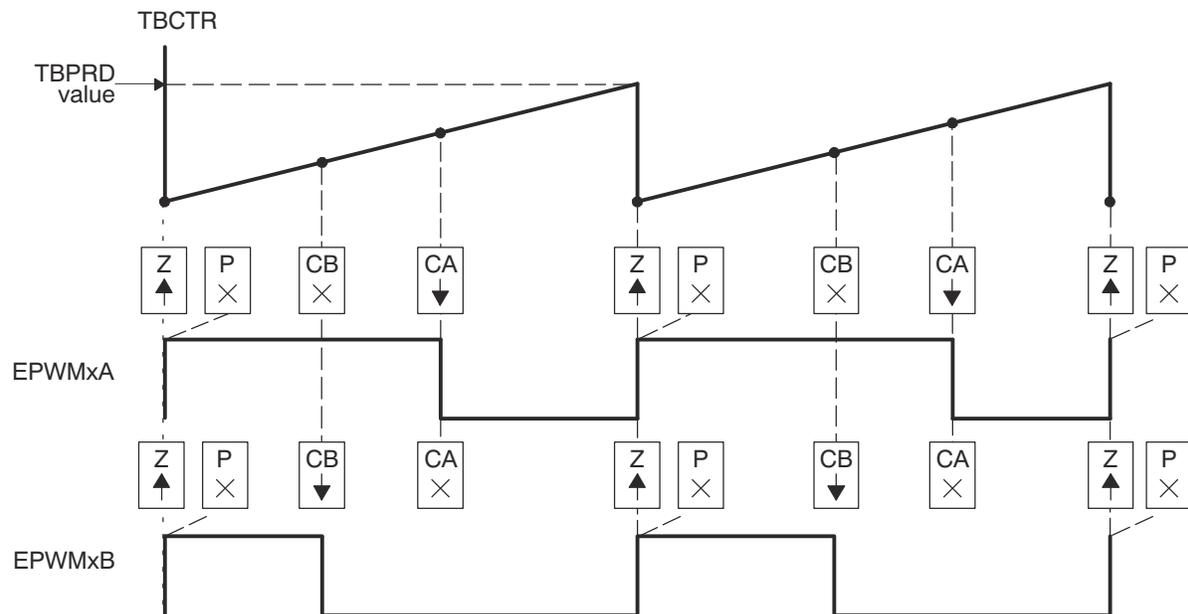
When using this configuration in practice, if loading CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If loading CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there is always a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.



**Figure 13-26. Up-Down Count Mode Symmetrical Waveform**

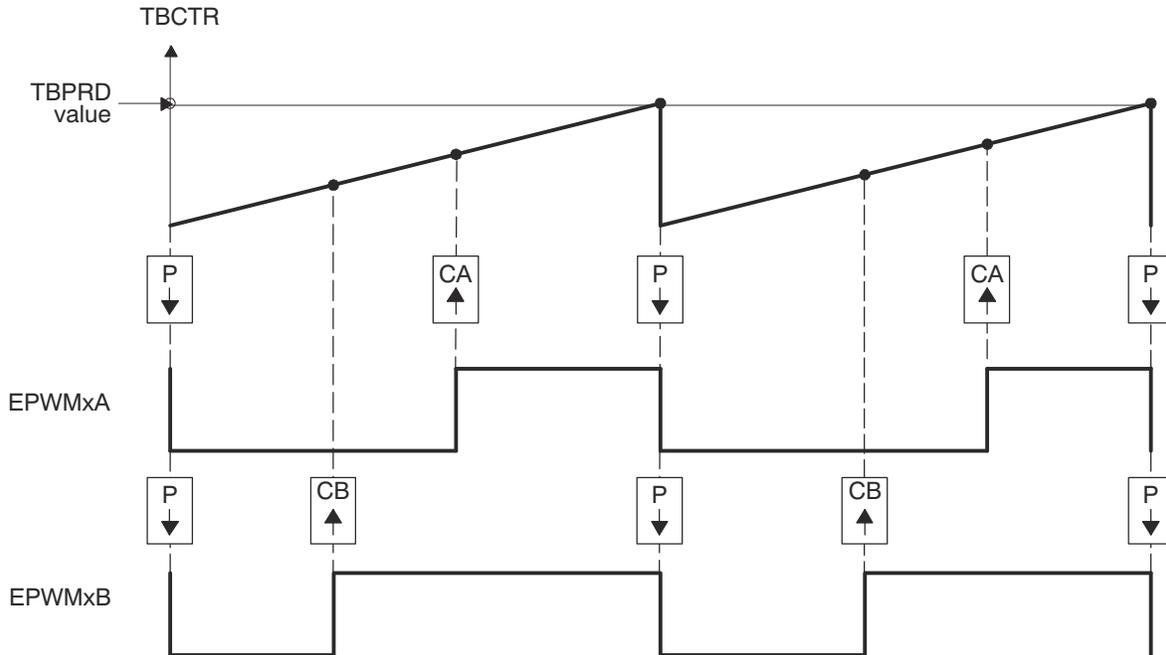
The PWM waveforms in [Figure 13-27](#) through [Figure 13-32](#) show some common action-qualifier configurations. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in the 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 count-down mode, Up means up-count mode and Down means down-count mode
- Sym = Symmetric, Asym = Asymmetric



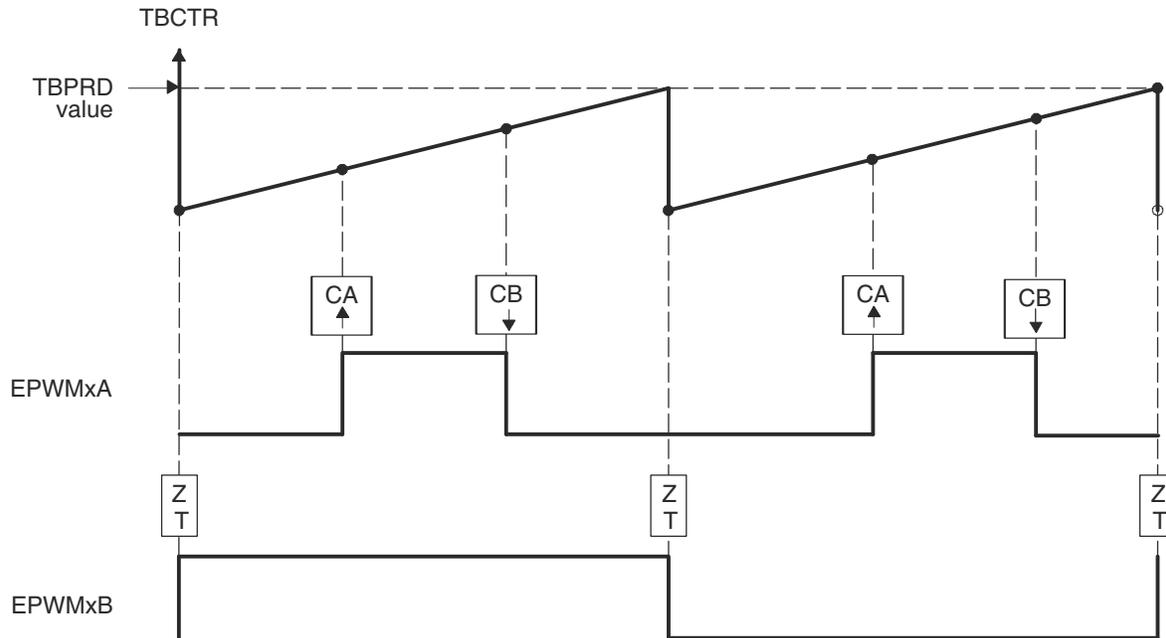
- PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
- Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
- The "Do Nothing" actions (X) are shown for completeness, but are not shown on subsequent diagrams.
- 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, Single Edge Asymmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB—Active High**



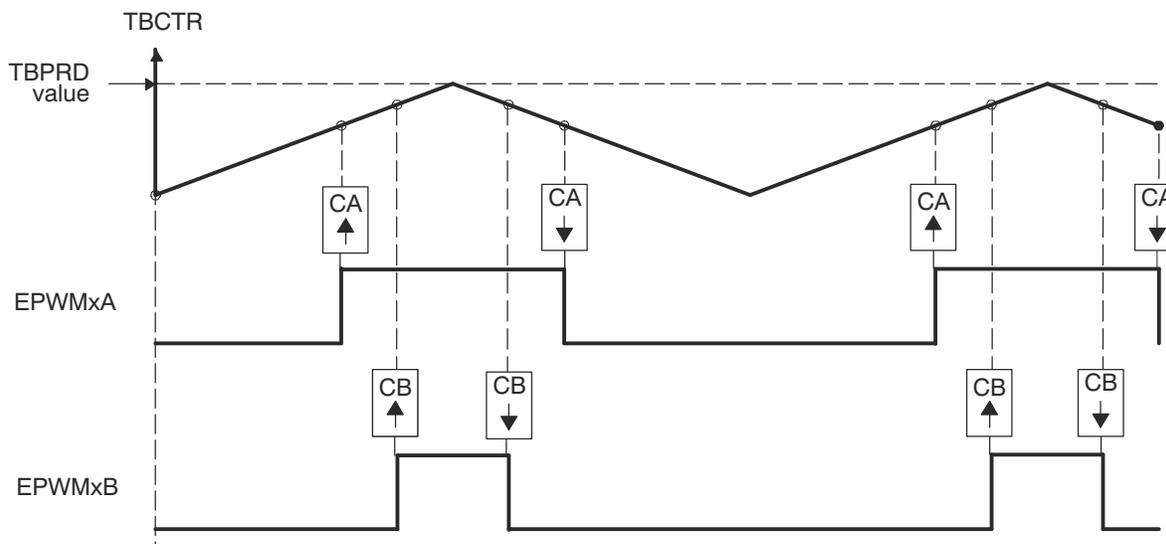
- PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 13-28. Up, Single Edge Asymmetric Waveform with Independent Modulation on EPWMxA and EPWMxB—Active Low**



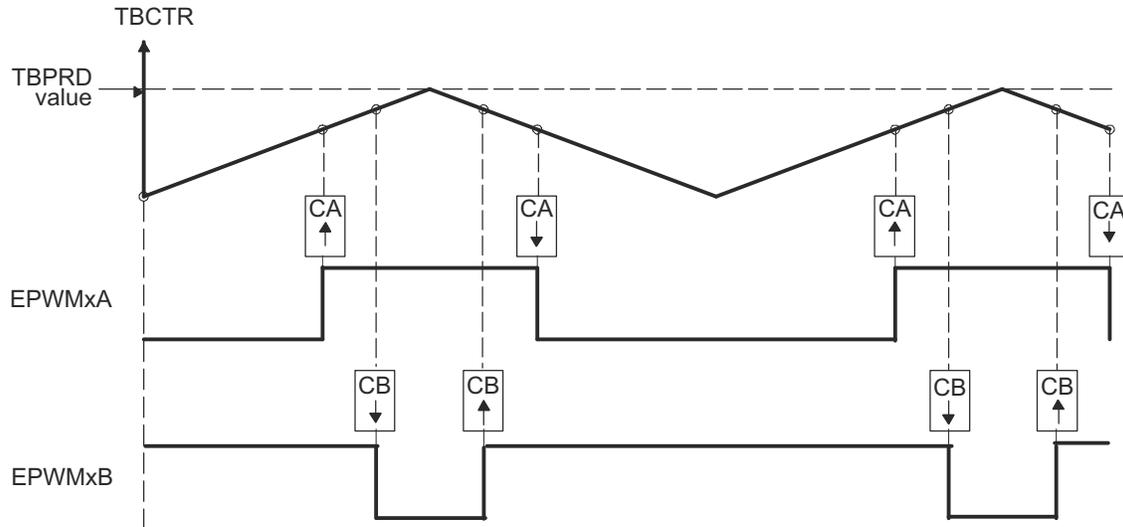
- A.  $\text{PWM frequency} = 1 / ((\text{TBPRD} + 1) \times T_{\text{TBCLK}})$
- B. Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)
- C. High time duty proportional to (CMPB - CMPA)

**Figure 13-29. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**



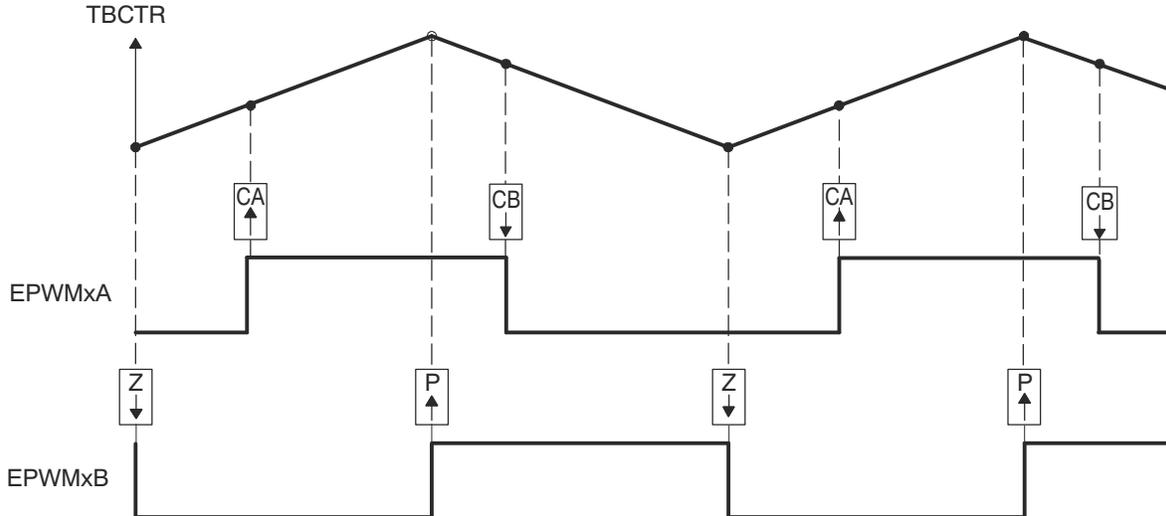
- A.  $\text{PWM period} = 2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D. Outputs EPWMxA and EPWMxB can drive independent power switches.

**Figure 13-30. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Active Low**



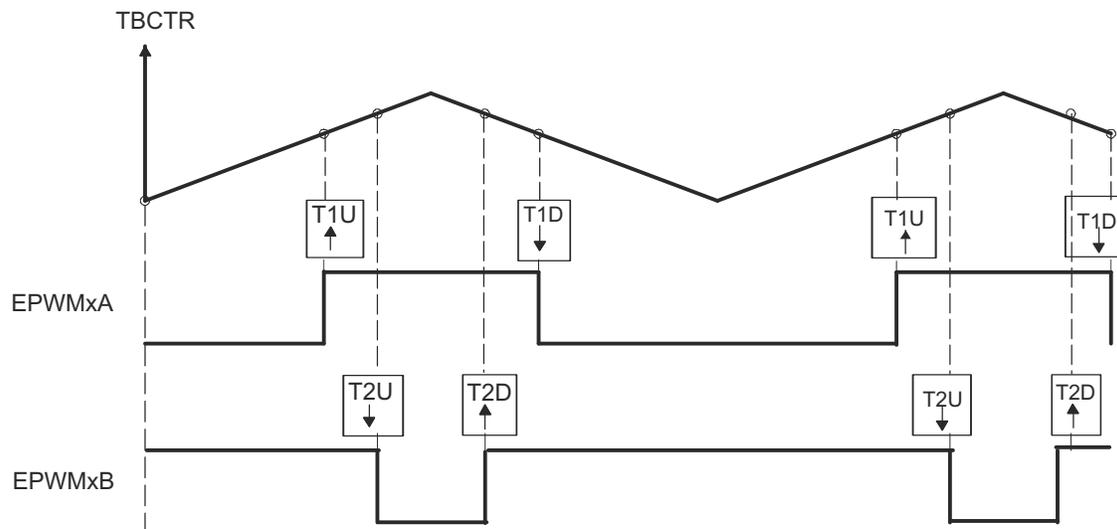
- A. PWM period =  $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B. Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
- C. Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
- D. Outputs EPWMx can drive upper/lower (complementary) power switches.
- E. Dead-band =  $\text{CMPB} - \text{CMPA}$  (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Figure 13-31. Up-Down Count, Dual-Edge Symmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB — Complementary**



- A. PWM period =  $2 \times \text{TBPRD} \times \text{TBCLK}$
- B. Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
- C. Duty modulation for EPWMxA is set by CMPA and CMPB.
- D. Low time duty for EPWMxA is proportional to  $(\text{CMPA} + \text{CMPB})$ .
- E. To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Clear on CMPA, Set on CMPB).
- F. Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB).

**Figure 13-32. Up-Down Count, Dual-Edge Asymmetric Waveform, with Independent Modulation on EPWMxA—Active Low**



- A.  $\text{PWM period} = 2 \times \text{TBPRD} \times \text{TTBCLK}$
- B. Independent T1 event actions when counter is counting up and when the counter is counting down are used to generate EPWMxA output.
- C. Independent T2 event actions when counter is counting up and when the counter 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.

**Figure 13-33. Up-Down Count, PWM Waveform Generation Utilizing T1 and T2 Events**

### 13.7 Dead-Band Generator (DB) Submodule

Figure 13-34 illustrates the dead-band submodule within the ePWM.

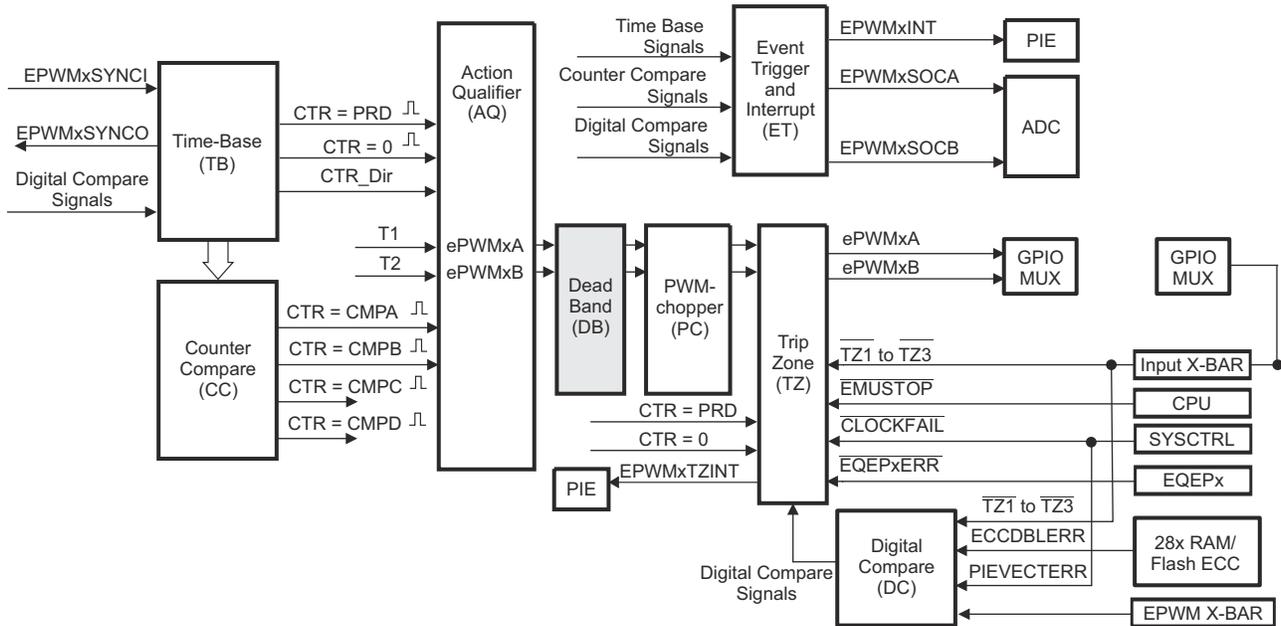


Figure 13-34. Dead\_Band Submodule

#### 13.7.1 Purpose of the Dead-Band Submodule

The action-qualifier (AQ) module section discussed how the AQ module can 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 must 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)

### 13.7.2 Dead-band Submodule Additional Operating Modes

On type 1 ePWM RED can appear on one channel output and FED can 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 13-35](#), 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
- Deadband and deadband high-resolution registers are now shadowed
- High-resolution deadband RED and FED have been enabled using the DBREDHR and DBFEDHR registers

---

#### Note

The PWM chopper is not enabled when high-resolution deadband is enabled.

High-resolution deadband 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.

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. Make sure the duty cycle value of the current waveform applied to the dead-band module is greater than the required phase shift amount.

The Type 4 action qualifier and dead-band outputs of the ePWM module are delayed by one TBCLK cycle in comparison to the Type 2 ePWM module, although the Type 4 behavior is the same as the Type 3 PWM. Both PWMA and PWMB signals are delayed under all circumstances.

---

### Shadow Mode:

The shadow mode for the DBRED is enabled by setting the DBCTL[SHDWDBREDDMODE] bit and the shadow register for DBFED is enabled by setting the DBCTL [SHDWDBFEDMODE] bit. Shadow mode is disabled by default for both DBRED and DBFED

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL [LOADREDDMODE] and 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

---

#### Note

The application software must enable shadow load mode in the DBCTL[SHDWDBREDDMODE] and DBCTL[SHDWDBFEDMODE] **before** programming values for the DBRED and DBFED registers. If the shadow register is enabled **after** programming the DBRED and DBFED registers, the DBRED and DBFED registers are loaded with a value of 0.

---

### Global Load Support

Global load control mechanism can also be used for DBRED:DBREDHR, DBFED:DBFEDHR, and DBCTL registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global load mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The Global load control mechanism is explained in [Section 13.4.7](#).

---

#### Note

When DBRED/DBFED active is loaded with a new shadow value while DB counters are counting, the new DBRED/DBFED value only affects the NEXT PWMx edge and not the current edge.

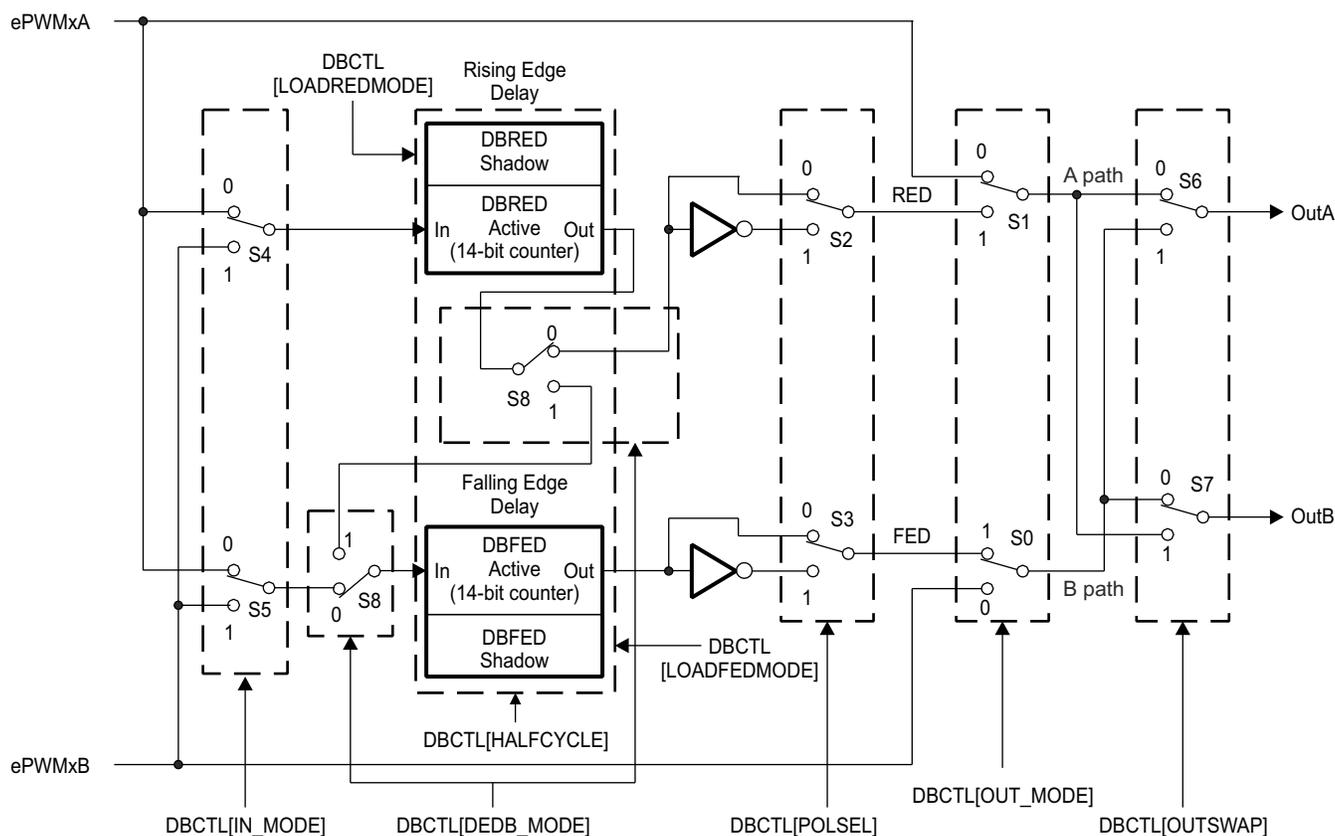
A Deadband value of zero cannot be used when the Global Shadow to Active Load is set to occur at CTR=ZERO. Similarly, a Deadband value of PRD cannot be used when the Global Shadow to Active Load is set to occur at CTR=PRD.

TBPRDHR cannot be used with Global load. If high-resolution period must be changed in the application, users must write to the individual period registers from an ePWM ISR (The ISR must be synchronous with the PWM switching period), where the Global Load One-Shot bit is also written to.

---

### 13.7.3 Operational Highlights for the Dead-Band Submodule

The configuration options for the dead-band submodule are shown in [Figure 13-35](#).



**Figure 13-35. Configuration Options for the Dead-Band Submodule**

Although all combinations are supported, not all are typical usage modes. [Table 13-9](#) 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 13-9](#) fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED):** Allows the user 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 can address all the active-high and active-low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in [Figure 13-36](#). Note that to generate equivalent waveforms to [Figure 13-36](#), configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge delay (RED) and Mode 7: Bypass falling-edge delay (FED):** Finally the last two entries in [Table 13-9](#) show combinations where either the falling-edge delay (FED) or rising-edge delay (RED) blocks are bypassed.

[Figure 13-36](#) shows waveforms for typical cases where  $0% < \text{duty} < 100%$ .

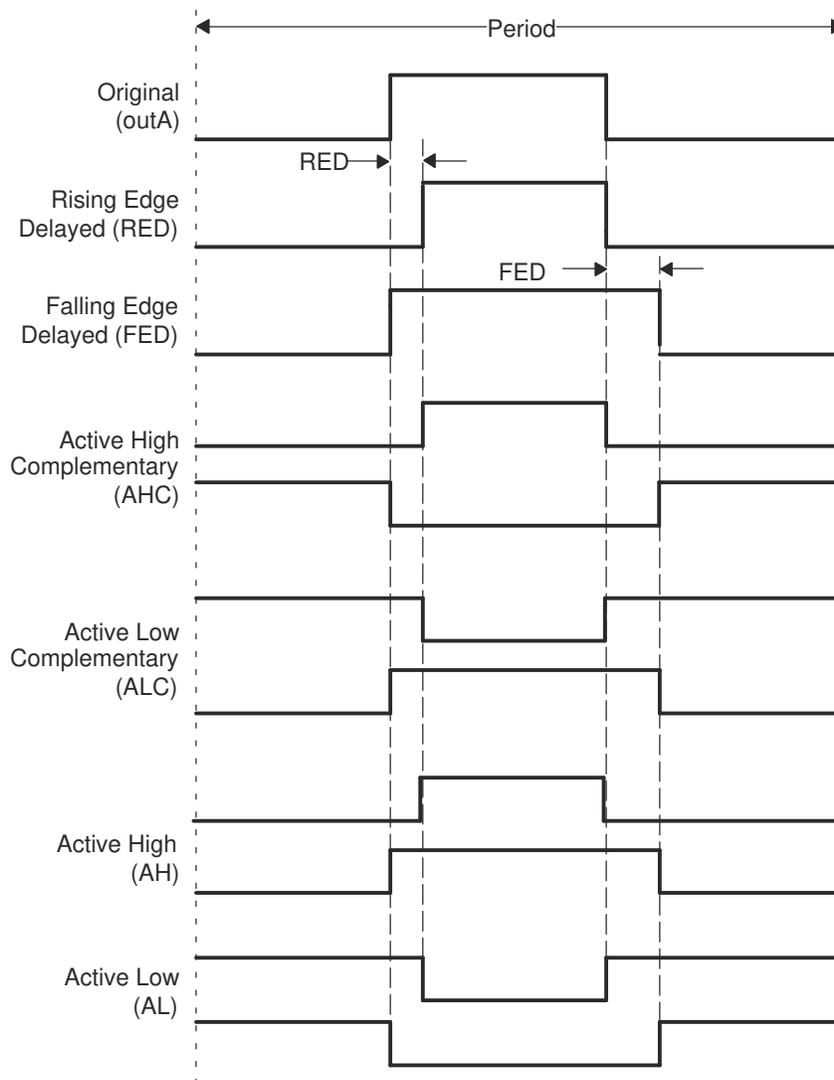
**Table 13-9. Classical Dead-Band Operating Modes**

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	X	X	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay)	0 or 1	0 or 1	0	1
	EPWMxB Out = EPWMxA In with Falling-Edge Delay				
7	EPWMxA Out = EPWMxA In with Rising-Edge Delay	0 or 1	0 or 1	1	0
	EPWMxB Out = EPWMxB In with No Delay				

**Table 13-10. Additional Dead-Band Operating Modes**

Mode Description	DBCTL[DEDB-MODE]	DBCTL[OUTSWAP]	
	S8	S6	S7
EPWMxA and EPWMxB signals are as defined by OUT-MODE bits.	0	0	0
EPWMxA = A-path as defined by OUT-MODE bits.	0	0	1
EPWMxB = A-path as defined by OUT-MODE bits (rising-edge delay or delay-bypassed A-signal path)			
EPWMxA = B-path as defined by OUT-MODE bits (falling-edge delay or delay-bypassed B-signal path)	0	1	0
EPWMxB = B-path as defined by OUT-MODE bits			
EPWMxA = B-path as defined by OUT-MODE bits (falling-edge delay or delay-bypassed B-signal path)	0	1	1
EPWMxB = A-path as defined by OUT-MODE bits (rising-edge delay or delay-bypassed A-signal path)			
Rising-edge delay applied to EPWMxA / EPWMxB as selected by S4 switch (IN-MODE bits) on A signal path only.	0	X	X
Falling-edge delay applied to EPWMxA / EPWMxB as selected by S5 switch (IN-MODE bits) on B signal path only.			
Rising-edge delay and falling-edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only. <sup>(1)</sup>	1	X	X

- (1) When this bit is set to 1, the user can always either set OUT\_MODE bits such that Apath = InA or set OUTSWAP bits such that EPWMxA=Bpath. Otherwise, EPWMxA is invalid.



**Figure 13-36. Dead-Band Waveforms for Typical Cases (0% < 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 the value represents the number of TBCLK (time-base clock) pulses 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  $T_{\text{TBCLK}}$  is the period of TBCLK, the prescaled version of EPWMCLK.

For convenience, delay values for various TBCLK options are shown in [Table 13-11](#). The ePWM input clock frequency that these delay values been computed by is 100MHz.

**Table 13-11. Dead-Band Delay Values in  $\mu$ s as a Function of DBFED and DBRED**

Dead-Band Value		Dead-Band Delay ( $\mu$ s)		
DBFED, DBRED	TBCLK = EPWMCLK/1	TBCLK = EPWMCLK /2	TBCLK = EPWMCLK/4	
1	0.01	0.02	0.04	
5	0.05	0.10	0.20	
10	0.10	0.20	0.40	
100	1.00	2.00	4.00	
200	2.00	4.00	8.00	
400	4.00	8.00	16.00	
500	5.00	10.00	20.00	
600	6.00	12.00	24.00	
700	7.00	14.00	28.00	
800	8.00	16.00	32.00	
900	9.00	18.00	36.00	
1000	10.00	20.00	40.00	

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

$$FED = DBFED \times T_{TBCLK}/2$$

$$RED = DBRED \times T_{TBCLK}/2$$

### 13.8 PWM Chopper (PC) 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 pulse transformer-based gate drivers to control the power switching elements are needed.

Figure 13-37 illustrates the PWM chopper submodule within the ePWM.

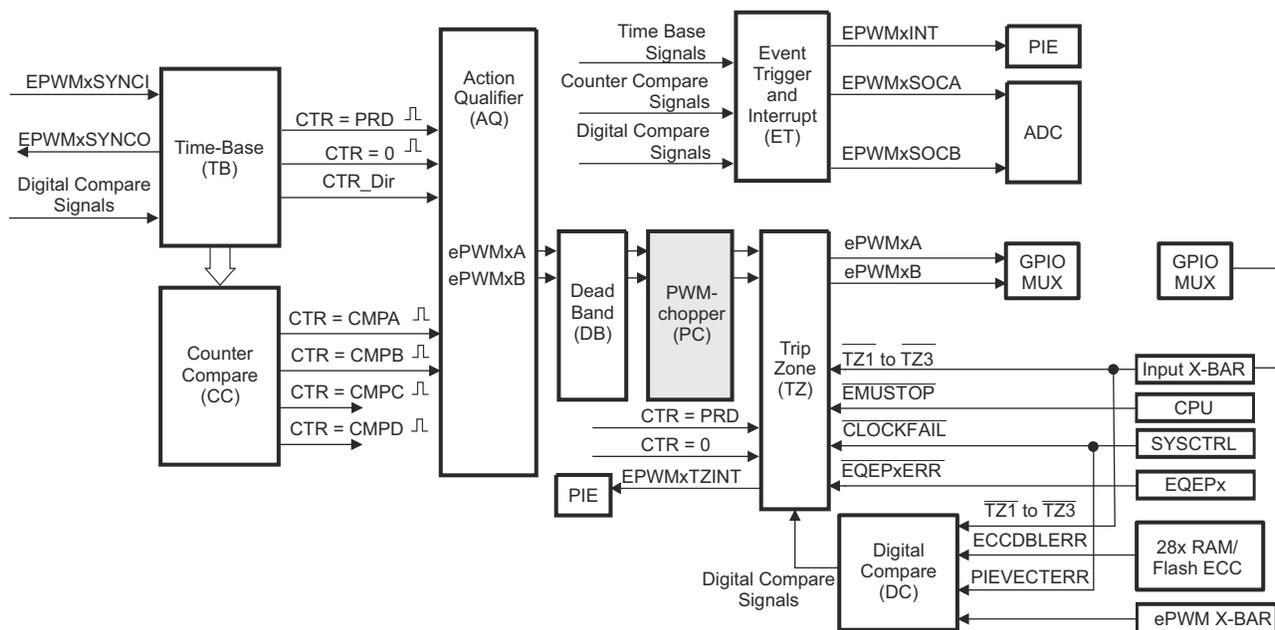


Figure 13-37. PWM Chopper Submodule

#### 13.8.1 Purpose of the PWM Chopper Submodule

The key functions of the PWM chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

#### 13.8.2 Operational Highlights for the PWM Chopper Submodule

Figure 13-38 shows the operational details of the PWM chopper submodule. The carrier clock is derived from EPWMCLK. The clock frequency and duty cycle are controlled using the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to make sure hard and fast power switch turn on, while the subsequent pulses sustain pulses, making sure the power switch remains on. The one-shot width is programmed using the OSHTWTH bits. The PWM chopper submodule can be fully disabled (bypassed) using the CHPEN bit.

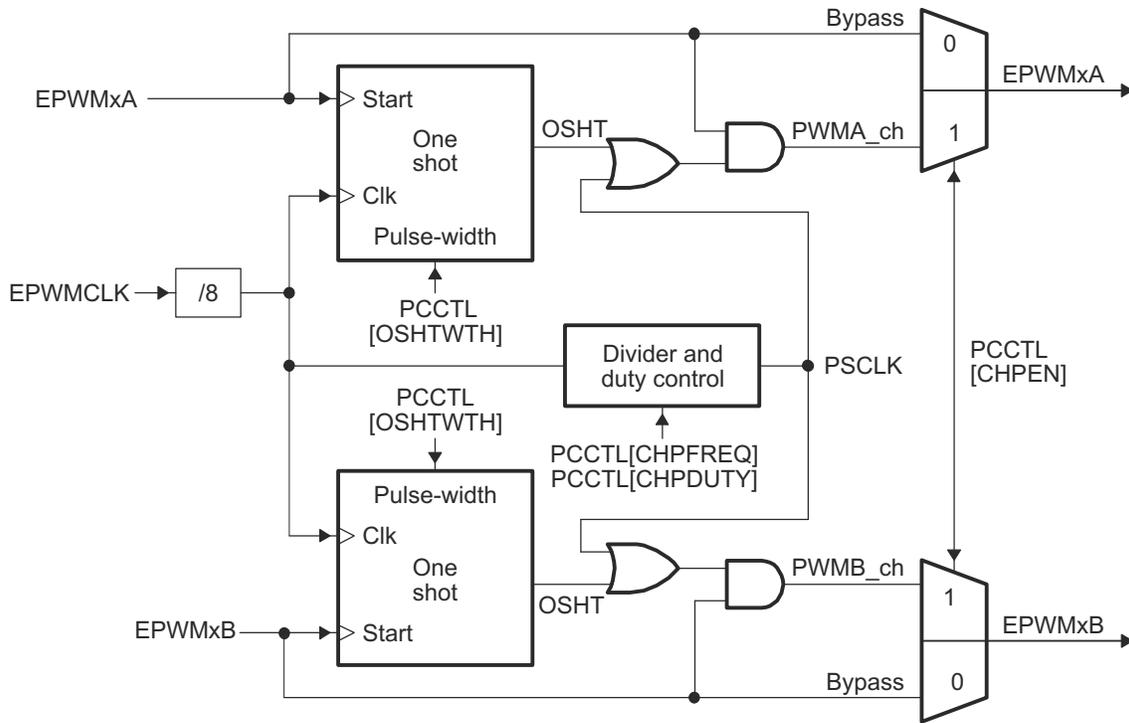


Figure 13-38. PWM Chopper Submodule Operational Details

### 13.8.3 Waveforms

Figure 13-39 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.

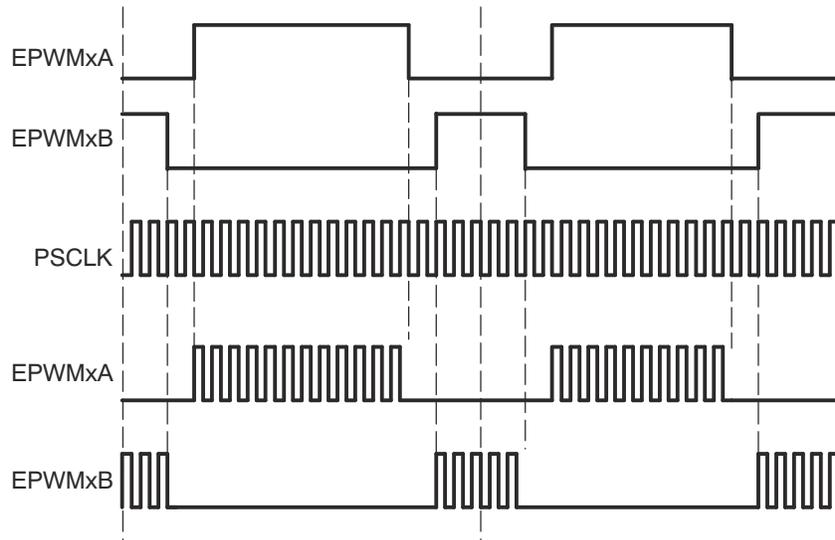


Figure 13-39. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only

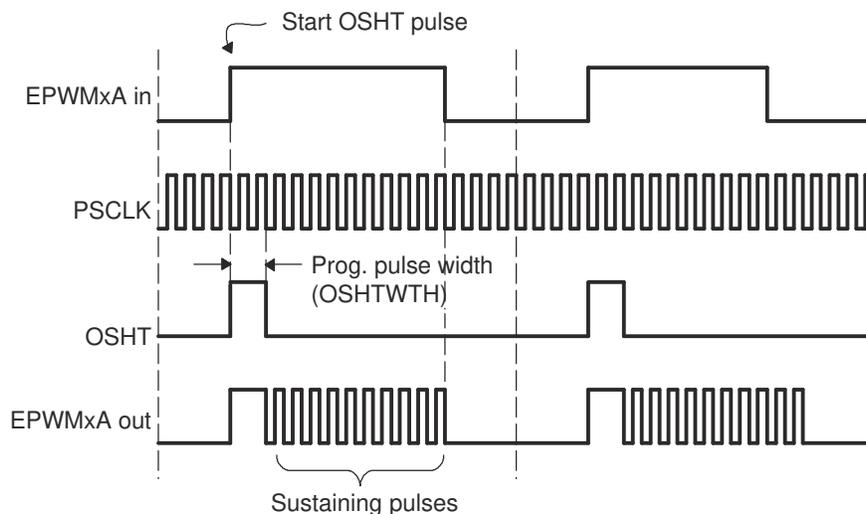
### 13.8.3.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1\text{stpulse}} = T_{\text{EPWMCLK}} \times 8 \times \text{OSHTWTH}$$

Where  $T_{\text{EPWMCLK}}$  is the period of the system clock (EPWMCLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 13-40 shows the first and subsequent sustaining pulses and Table 13-12 gives the possible pulse width values for a EPWMCLK = 80MHz.



**Figure 13-40. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

**Table 13-12. Possible Pulse Width Values for EPWMCLK = 80MHz**

OSHTWTH (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

### 13.8.3.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses make sure 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 using software control.

Figure 13-41 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%.

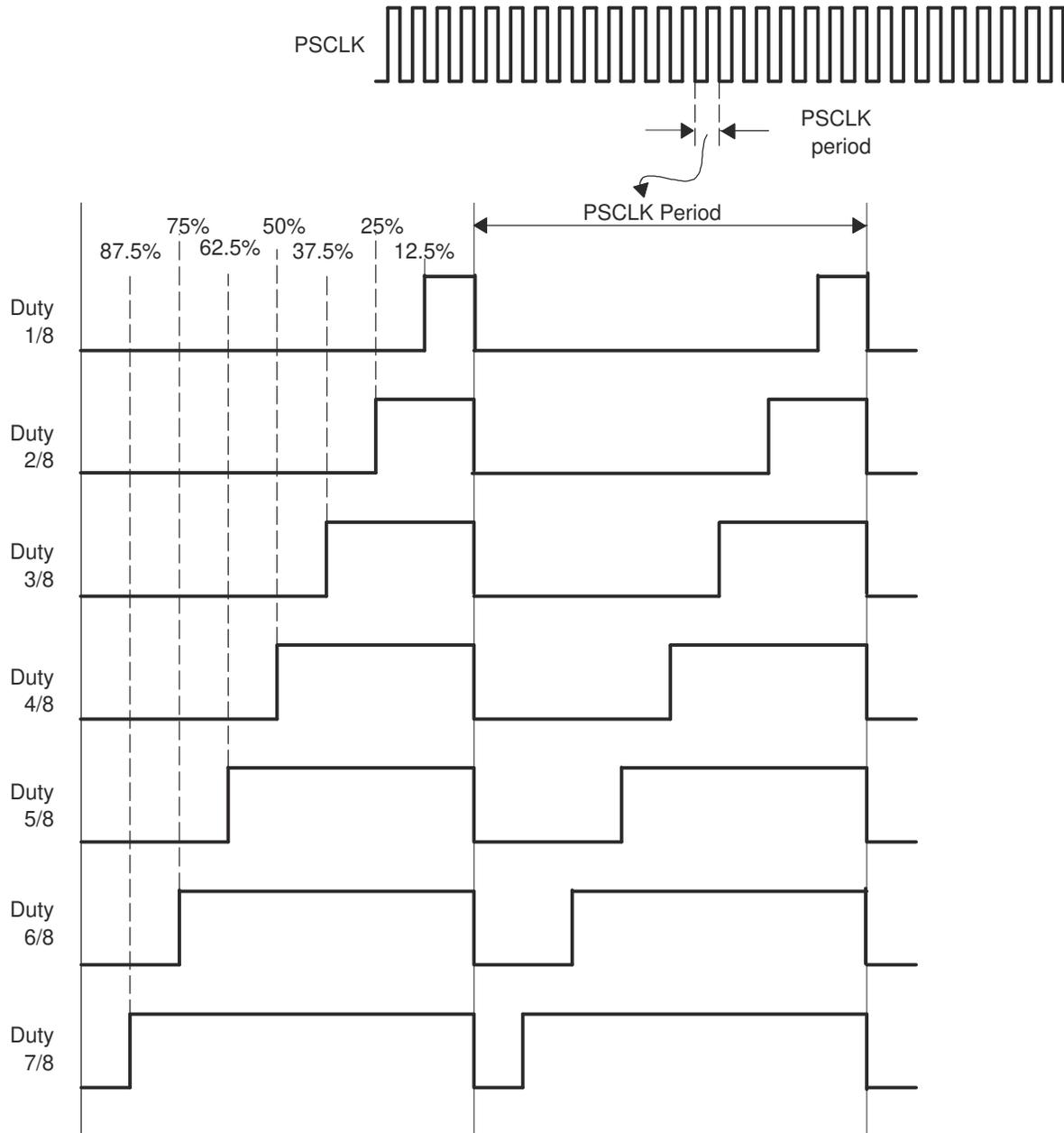
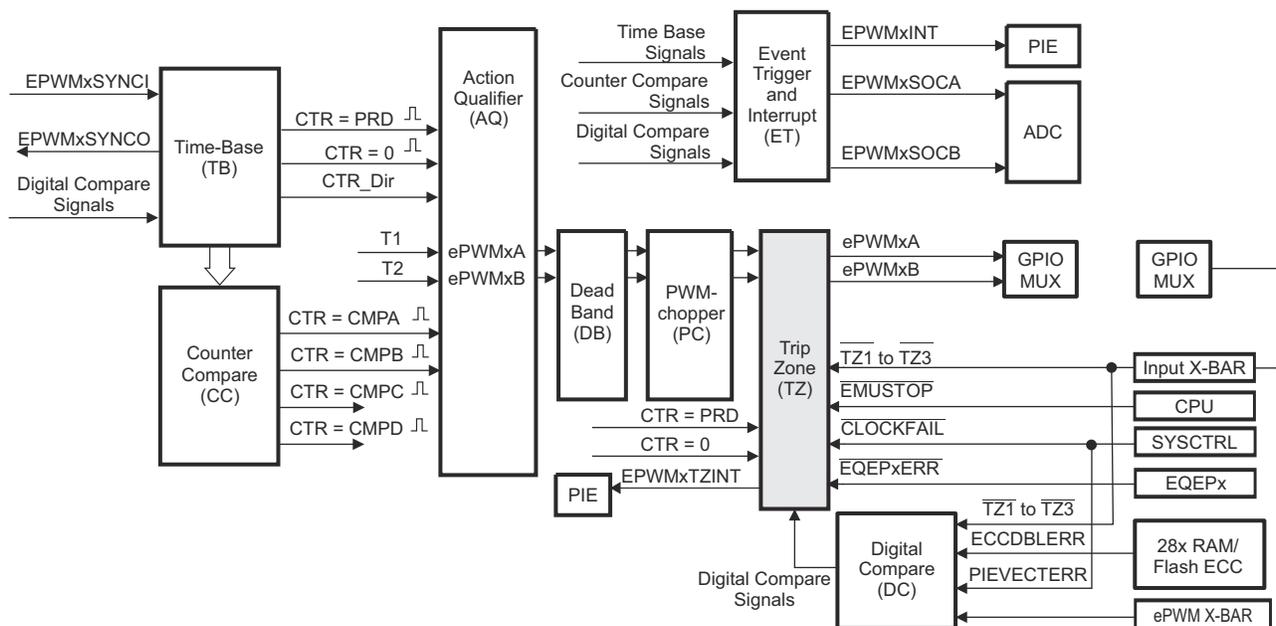


Figure 13-41. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses

## 13.9 Trip-Zone (TZ) 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.

Figure 13-42 illustrates the trip-zone submodule within the ePWM.



**Figure 13-42. Trip-Zone Submodule**

### 13.9.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs  $\overline{TZ1}$  to  $\overline{TZ6}$  can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Support for digital compare tripping (DC) based on state of on-chip analog comparator module outputs and  $\overline{TZ1}$  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 the trip-zone submodule is not required.

### 13.9.2 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals  $\overline{TZ1}$  to  $\overline{TZ6}$  (also collectively referred to as  $\overline{TZn}$ ) are active-low input signals. When one of these signals goes low, or when a DCAEVT1/2 or DCBEVT1/2 force happens based on the TZDCSEL register event selection, the indication is 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 can or cannot be synchronized to the ePWMclock (EPWMCLK) and digitally filtered within the GPIO MUX block. A minimum of  $3 \cdot TBCLK$  low pulse width on  $\overline{TZn}$  inputs is sufficient to trigger a fault condition on the ePWM module. If the pulse width is less than this, the trip condition cannot 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 *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 EPWMxB outputs. [Table 13-13](#) lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up or while the counter 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 when enabled in the TZEINT register and interrupt controller. A corresponding flag for the event that caused the CBC event is also set in register TZCBCFLG.

If the CBC interrupt is enabled using the TZEINT register and DCAEVT2 or DCBEVT2 are selected as CBC trip sources using 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 remain set until the flag bits 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 TZCBCFLG register bits are cleared, then these bits are again 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 EPWMxB output. [Table 13-13](#) 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 while the counter 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 when enabled in the TZEINT register and interrupt controller. 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, the TZOSTFLG register bit can be cleared by manually writing to the corresponding bit in the TZOSTCLR register.

If the one-shot interrupt is enabled using the TZEINT register and DCAEVT1 or DCBEVT1 are selected as OSHT trip sources using 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.

---

**Note**

Clear the TZFLG and TZOSTFLG flags after making sure that the TRIPIN source of the OST has become inactive. Otherwise, if interrupts are enabled, depending on when the flags are cleared, an OST interrupt can occur and the OST flags are zero.

---

- **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 using the DCTRIPSEL register and can be either trip zone input pins or analog comparator CMPSSx signals. For more information on the digital compare submodule signals, see [Section 13.11](#).

When a digital compare event occurs, the action specified in the TZCTL[DCAEVT1/2] and TZCTL[DCBEVT1/2] bits is carried out immediately on the EPWMxA and EPWMxB output. [Table 13-13](#) lists the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and while the counter is counting down by appropriately configuring bits in TZCTLDCA and TZCTLDCB register. Actions specified in TZCTLDCA and TZCTLDCB registers take effect only when ETZE bit in TZCTL2 is set.

In addition, the relevant DC trip event flag (TZFLG[DCAEVT1/2] / TZFLG[DCBEVT1/2]) is set and a EPWMx\_TZINT interrupt is generated when enabled in the TZEINT register and interrupt controller.

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 remains set until the flag 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 the flag is again immediately set.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL, TZCTL2, TZCTLDCA, and TZCTLDCB register bit fields. Some of the possible actions, shown in [Table 13-13](#), can be taken on a trip event.

**Table 13-13. Possible Actions On a Trip Event**

TZCTL Register Bitfield Settings	EPWMxA and 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 is forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B is 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 is forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B is 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 cycle-by-cycle event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A is forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B is 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
  - TZSEL[OSHT6] = 1: enables  $\overline{TZ6}$  as a one-shot event source for ePWM2
  - TZCTL[TZA] = 0: EPWM2A is put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM2B ignores the trip event.

#### Note

When configuring the GPIOs and INPUT X-BAR/EPWM X-BAR options, be aware that a change in the X-BAR input selections can cause an unwanted event. Therefore, set up the GPIO and X-BAR input configurations before enabling the ePWM Trip-Zone. If a requirement is to change the GPIO/X-BAR configurations while the ePWM Trip-Zone is enabled, the user can turn off the TRIPs by clearing the TZSEL register and reconfiguring the TRIP selection (TZSEL) after the INPUT XBAR selection is changed.

### 13.9.3 Generating Trip Event Interrupts

Figure 13-43 and Figure 13-44 illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in Section 13.11.





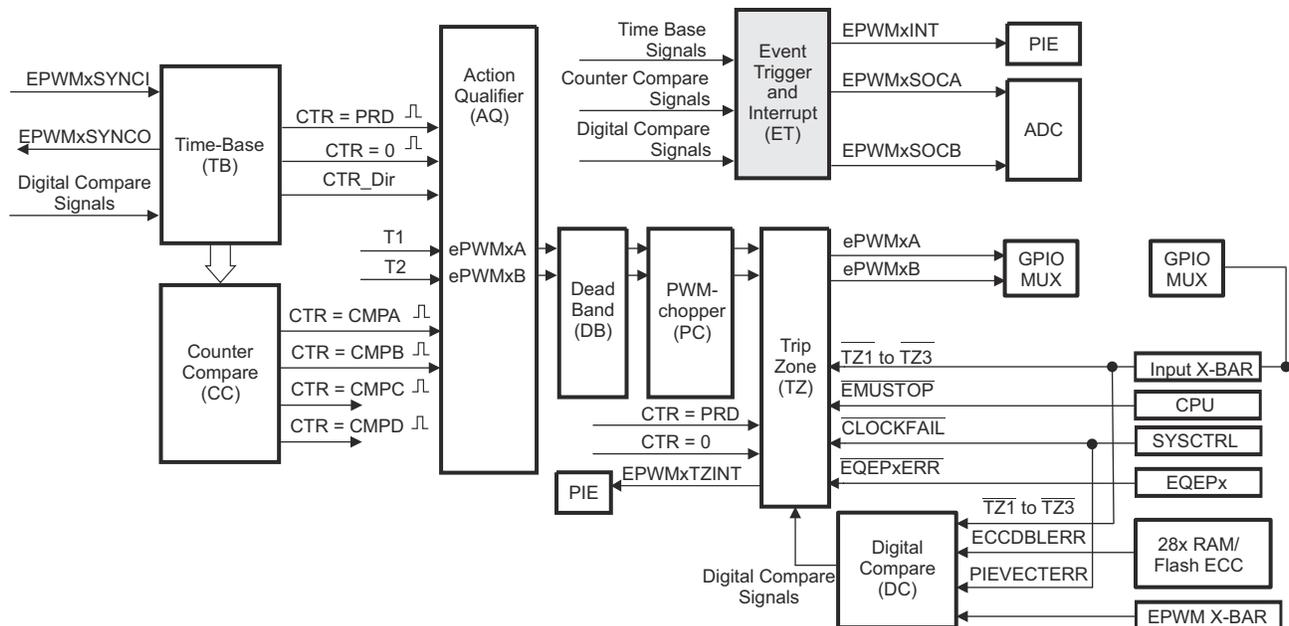
### 13.10 Event-Trigger (ET) Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base, counter-compare, and digital-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Up to every fifteenth event
- Provides full visibility of event generation using 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 a start of conversion pulse to the ADC when a selected event occurs.

Figure 13-45 illustrates the event-trigger submodule within the ePWM.

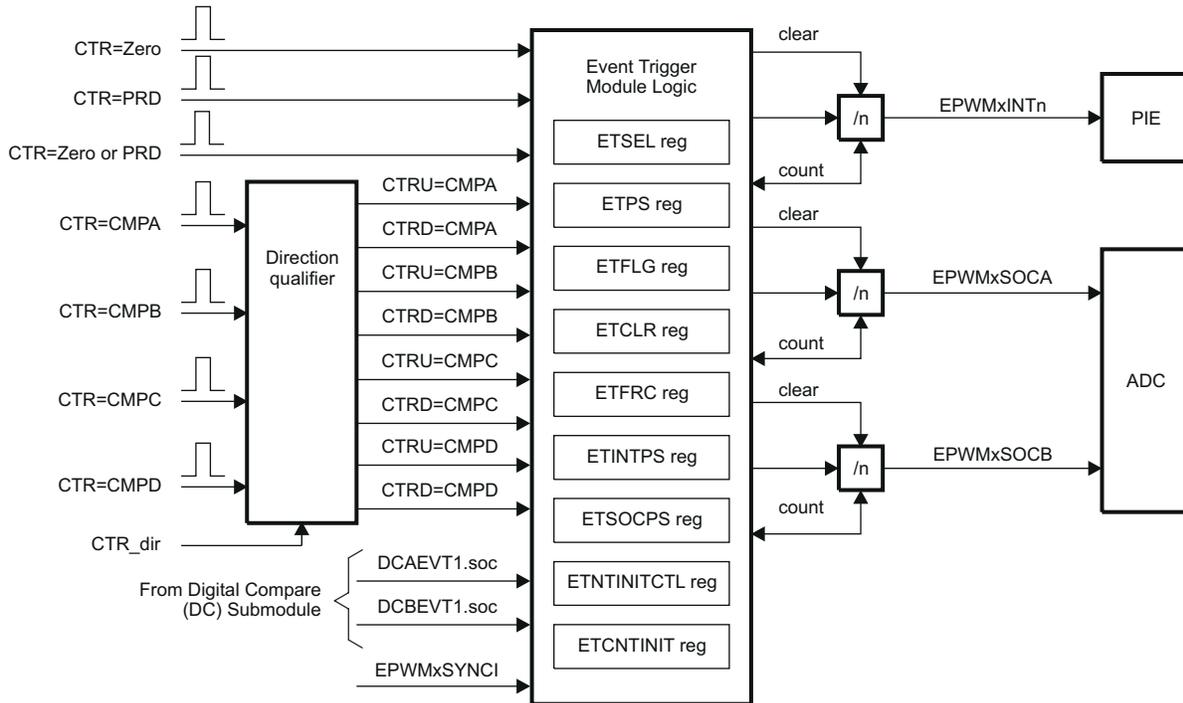


**Figure 13-45. Event-Trigger Submodule**

### 13.10.1 Operational Overview of the ePWM Event-Trigger Submodule

The event-trigger submodule monitors various event conditions (shown as inputs on the left side of Figure 13-46) 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 13-46. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**

- ETSEL - This selects which of the possible events 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 clearing the flag bits in the ETFLG register using 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 using SYNC event or using software force.
- ETCNTINIT - These bits allow initializing 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 13-47, Figure 13-48, and Figure 13-49.

Figure 13-47 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.

The selection made on ETPS[INTPSEL] bit determines whether ETINTPS register, INTCNT2 and INTPRD2 bit fields determine frequency of events (interrupt once every 0-15 events).

The event that 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[INTPSEL]. That is, when the specified event occurs the ETPS[INTCNT] or ETINTPS[INTCNT2] bits are incremented until the bits reach the value specified by ETPS[INTPRD] or ETINTPS[INTPRD2] determined again by the selection made in ETPS[INTPSEL]. When ETPS[INTCNT] = ETPS[INTPRD], the counter stops counting and the counter output is set. The counter is only cleared when an interrupt is sent to the interrupt controller.

When ETPS[INTCNT] reaches ETPS[INTPRD], the following behavior occurs. [The following behavior is also applicable to ETINTPS[INTCNT2] and 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 begins 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 the counter reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter holds the output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing a 0 to the INTPRD bits automatically clears the counter (INTCNT = 0) and the counter output resets (so no interrupts are generated). For all other writes to INTPRD, INTCNT retains the previous value. INTCNT resets when INTCNT overflows. Writing a 1 to the ETFRC[INT] bit increments the event counter INTCNT. The counter behaves as previously described when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events are detected and the ETFRC[INT] bit is also ignored. The same applies to ETINTPS[INTCNT2] and ETINTPS[INTPRD2].

The previous definition means that an interrupt on every event, on every second event, or on every third event if using the INTCNT and INTPRD can be generated. An interrupt on every event up to 15 events if using the INTCNT2 and INTPRD2 can be generated.

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 initialization of INTCNT2 counter with contents of ETCNTINIT[INTINIT] on a SYNC event or software force is determined by ETCNTINITCTL[INTINITFRC].

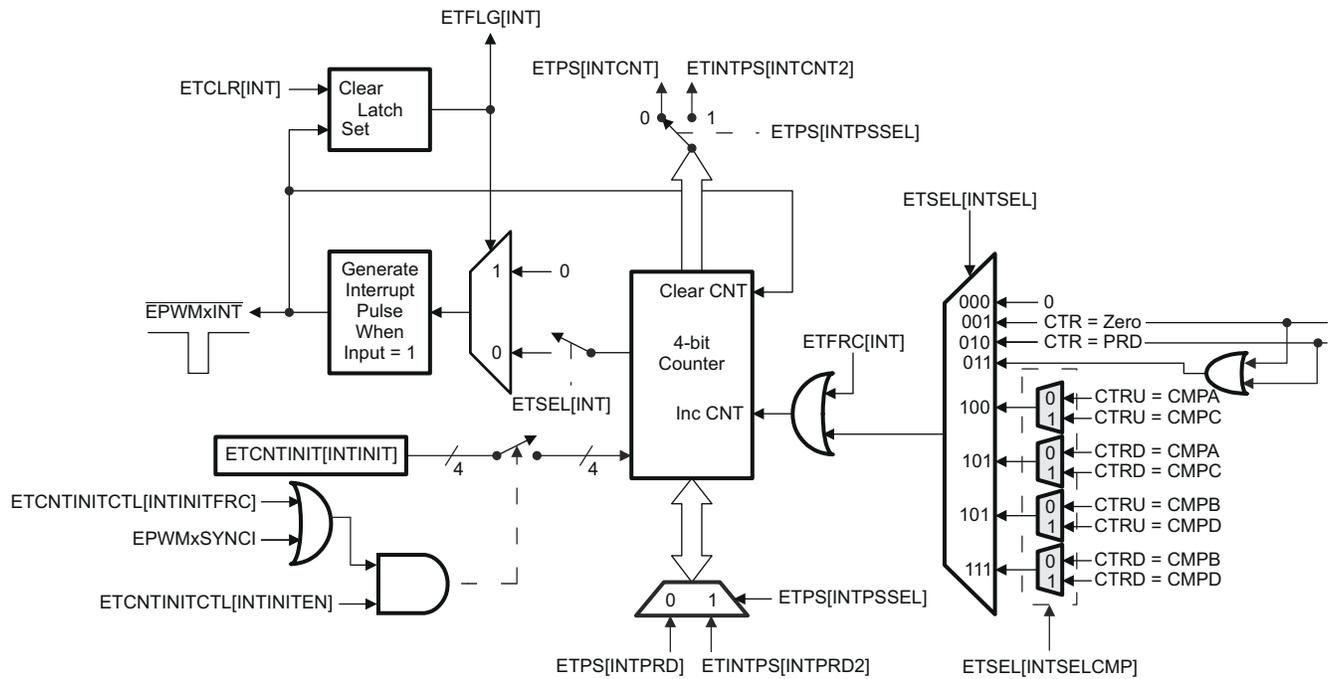
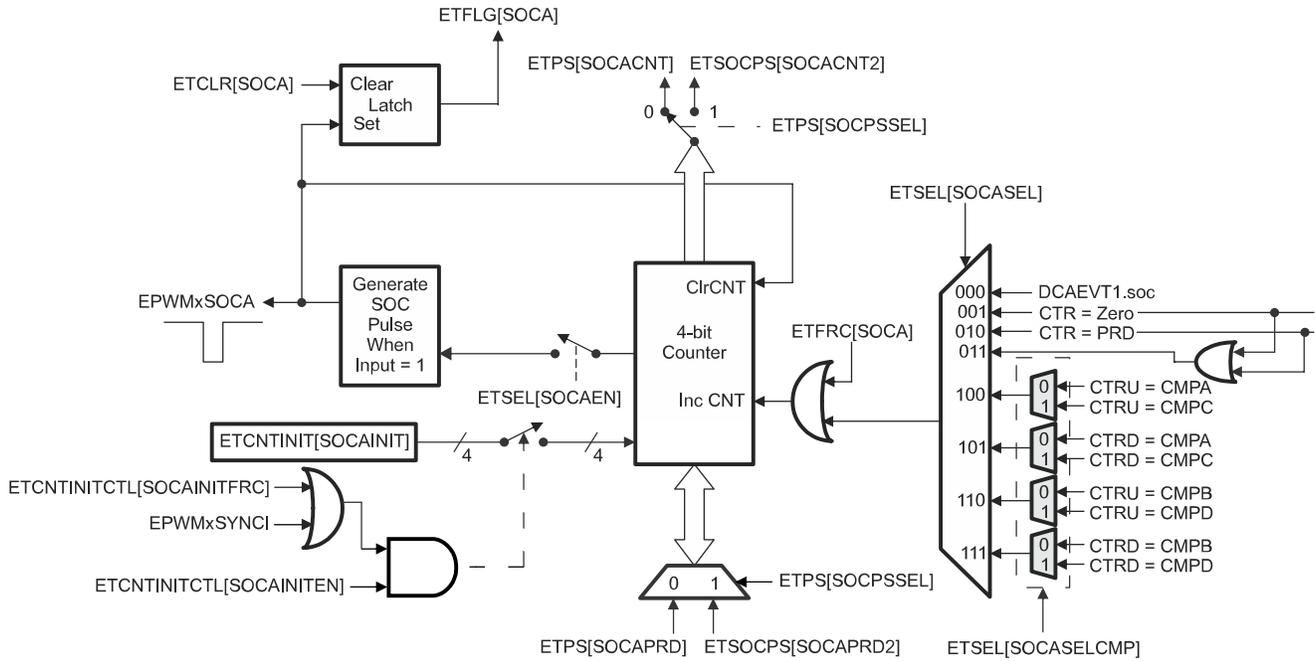


Figure 13-47. Event-Trigger Interrupt Generator

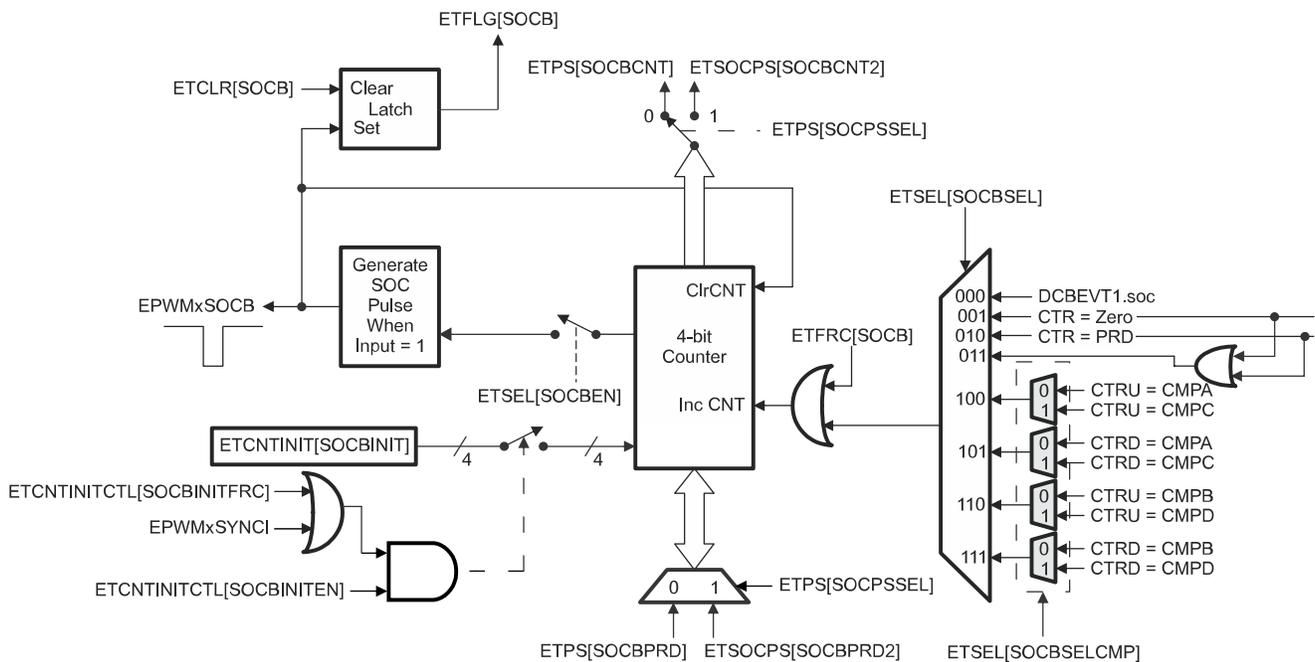
Figure 13-48 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The enhancements include SOCASELCMP and SOCBSELCMP bit fields defined in the ETSEL register enable CMPC and CMPD events respectively to cause a start of conversion. The ETPS[SOCPSSEL] bit field determines 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 the interrupt generator does not stop further pulse generation. The enable and disable bit ETSEL[SOCASEN] 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 triggers 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.



NOTE: The DCAEVT1.soc signals are generated by the Digital Compare (DC) submodule in [Section 13.11](#).

**Figure 13-48. Event-Trigger SOCA Pulse Generator**

[Figure 13-49](#) 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.



NOTE: The DCBEVT1.soc signals are generated by the Digital Compare (DC) submodule in [Section 13.11](#).

**Figure 13-49. Event-Trigger SOCB Pulse Generator**

### 13.11 Digital Compare (DC) Submodule

Figure 13-50 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

The eCAP input signals are sourced from the Input X-BAR signals as shown in Figure 13-51.

On this device, any of the GPIO pins can be flexibly mapped to be the trip-zone input and 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 the correct state on the selected pin before enabling the clock and configuring the trip input for the respective ePWM peripheral to avoid spurious latch of the TRIP signal.

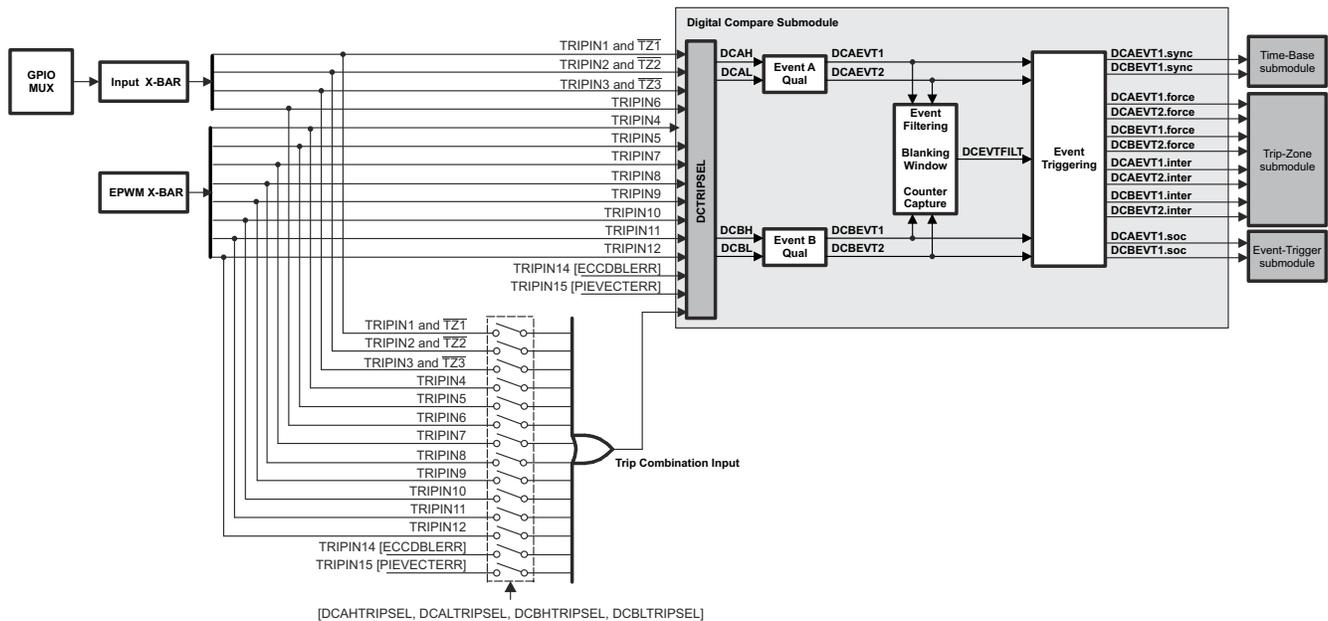


Figure 13-50. Digital-Compare Submodule High-Level Block Diagram

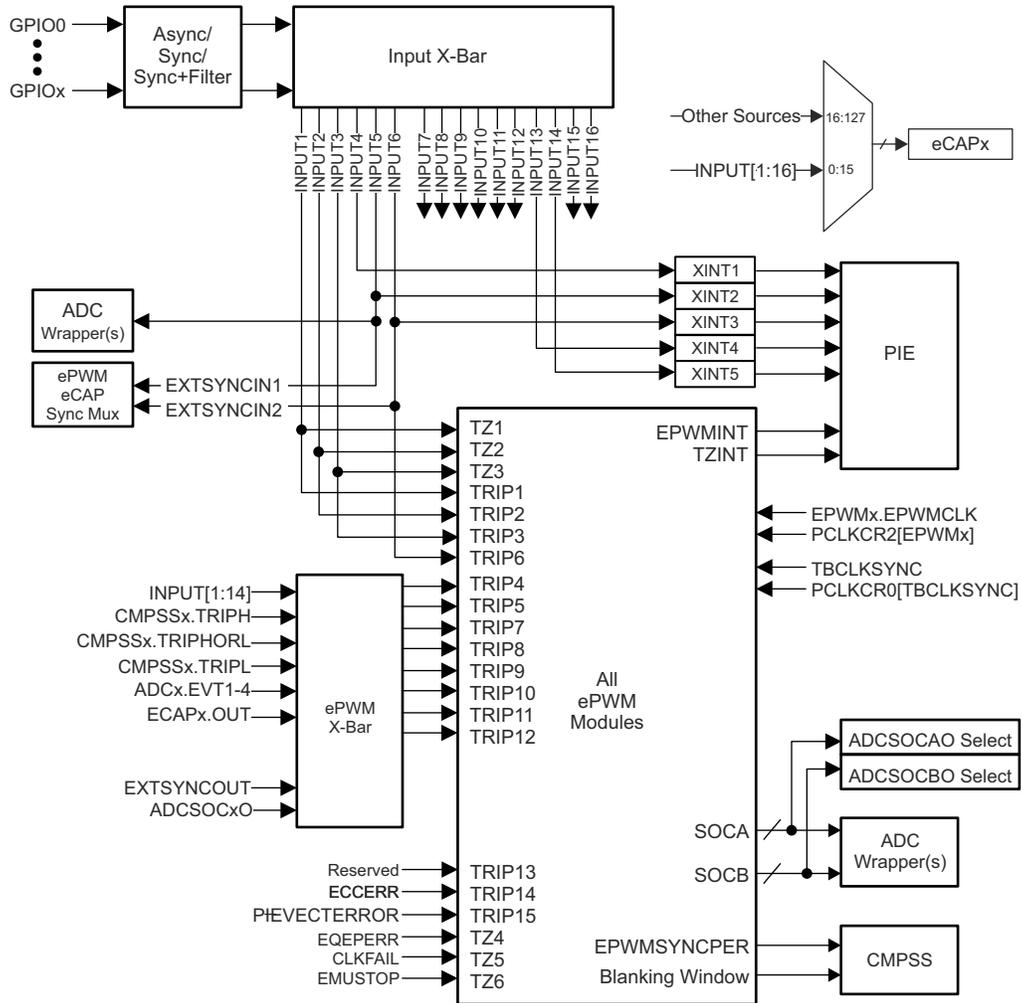


Figure 13-51. GPIO MUX-to-Trip Input Connectivity

### 13.11.1 Purpose of the Digital Compare Submodule

The key functions of the digital compare submodule are:

- Analog comparator (COMP) module outputs fed through the Input X-BAR, EPWM X-BAR, externally using the GPIO peripheral, interrupt controller signals, ECC error signals, TZ1,  $\overline{TZ2}$ , and  $\overline{TZ3}$  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 that can then either be filtered or applied 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.

### 13.11.2 Enhanced Trip Action Using CMPSS

To allow multiple CMPSS at a time to affect DCA/BEVTx events and trip actions, there is a OR logic to bring together ALL trip inputs (up to 15) from sources external to the ePWM module and feed into DCAH, DCAL, DCBH, and DCBL as a “combinational input” using the DCTRIPSEL register. This is configured by selecting “Trip combination input” (value of 0xF) in the DCTRIPSEL register.

There is a discrete choice of which trip inputs to put through the combinational logic for generating the DCAH, DCAL, DCBH, and DCBL signals. This is achieved using the DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL, and DCBLTRIPSEL register selections. Inputs selected for combinational input are passed through to the DCTRIPSEL register.

### 13.11.3 Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis

When using the CMPSS to trip the ePWM on a cycle-by-cycle basis, steps can 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 are 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 remains 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 is not cleared until after the following PWM cycle has started. Thus, the new PWM cycle detects a trip condition as soon as the cycle begins.

To avoid this undesired trip condition, the application can take steps to make sure 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 is not asserted within N clock cycles prior to the end of the PWM cycle.
- Activate blanking of the comparator trip signal using the ePWM event filter at least two clock cycles prior to the PWMSYNCPER signal and continue blanking for at least N clock cycles into the next PWM cycle.
- If the CMPSS COMPxLATCH path is used, clear the COMPxLATCH at least N clock cycles prior to the end of the PWM cycle. The latch can be cleared by software (using COMPSTSCLR) or by generating an early PWMSYNCPER signal. The ePWM modules on this device include the ability to generate PWMSYNCPER upon a CMPC or CMPD match (using HRPCTL) for arbitrary PWMSYNCPER placement within the PWM cycle.

### 13.11.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

#### 13.11.4.1 Digital Compare Events

As described in [Section 13.11.1](#), trip zone inputs ( $\overline{TZ1}$ ,  $\overline{TZ2}$ , and  $\overline{TZ3}$ ) and CMPSSx signals from the analog comparator (COMP) module can be selected using 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 TZDCSEL 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  $3 \times TBCLK$  sync pulse width is required. If pulse width is  $< 3 \times TBCLK$  sync pulse width, the trip condition can or can 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 [Event Filtering](#). 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 (using TZCTL, TZCTLDCA, TZCTLDCB register configurations) or, if the DCAEVT1/2 signals are selected as one-shot or cycle-by-cycle trip sources (using the TZSEL register), the DCAEVT1/2.force signals can effect the trip action using 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 interrupt controller. To enable the interrupt, 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 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 using 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 using 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.

Figure 13-52 and Figure 13-53 show how the DCxEVT1, DCxEVT2, or DCEVTFLT signals are processed to generate the digital compare A and B event force, interrupt, soc and sync signals.

In some of the applications like Phase Shifted Full Bridge (PSFB) Converters, it is required that different actions are taken on a CBC trip event and an OST trip event. This can be achieved using the DCxEVT1LAT.

- This latch can be cleared on CNT=0, CTR=PRD, and CNT=0 OR CTR=PRD events based on the setting of DCxCTL.EVTy.LATCLRSEL setting. This is similar to CBC latch clear mechanism.
- DCxEVTy.force signal can be chosen to be either the latched version or the unlatched version based on DCxCTL.EVTyLATSEL value.
- The status of DCxEVTyLAT signal can be accessed by reading DCxCTL.EVTyLAT field.

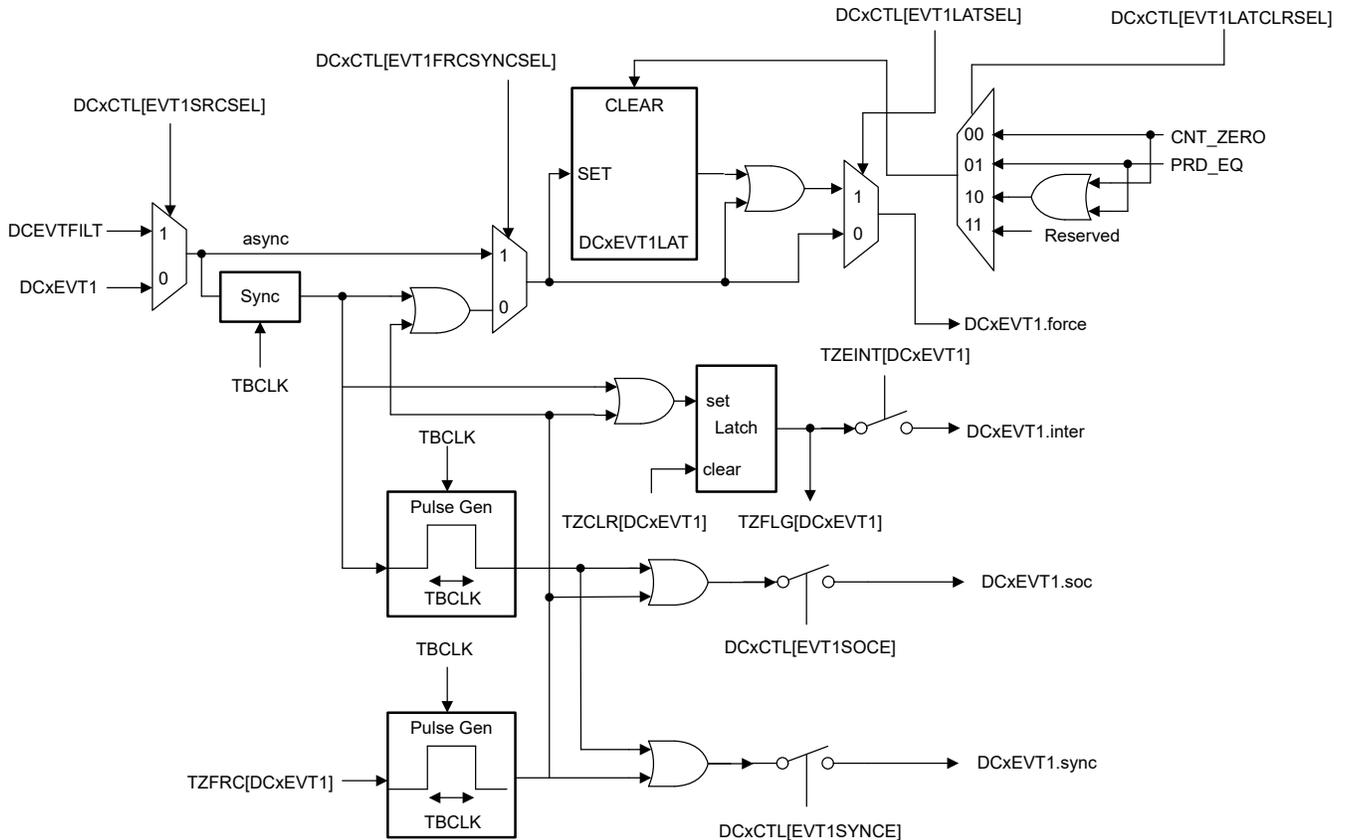


Figure 13-52. DCxEVT1 Event Triggering

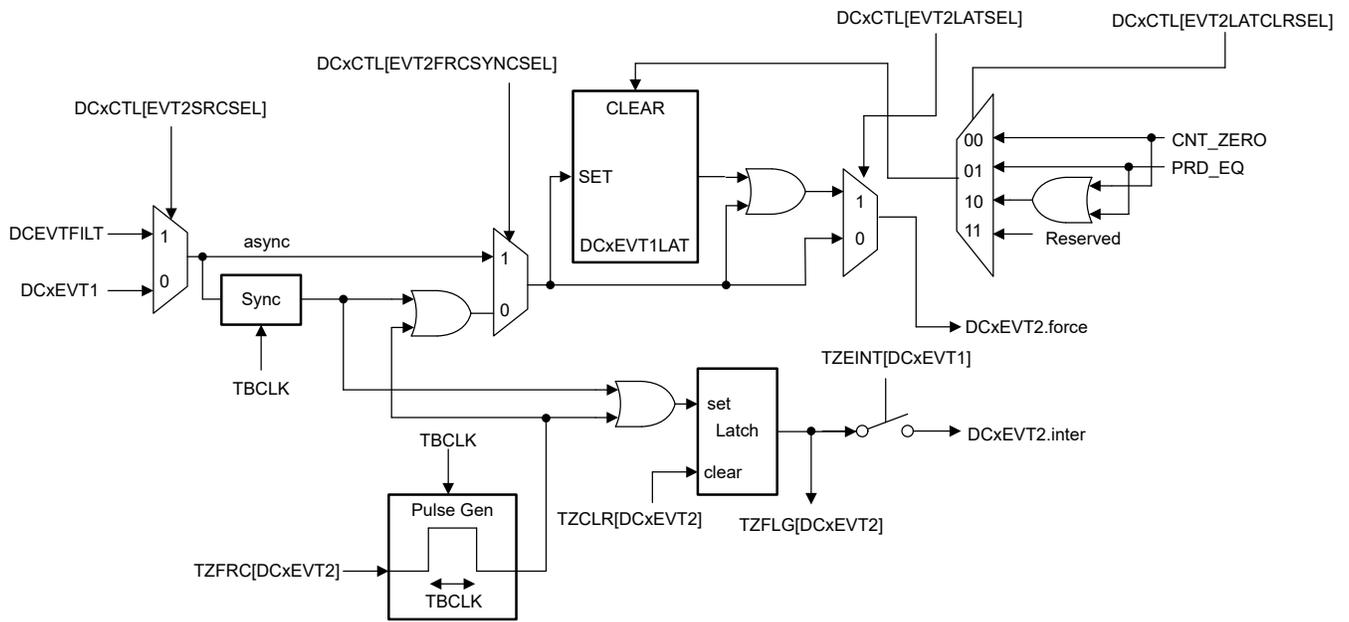


Figure 13-53. DCxEVT2 Event Triggering

### 13.11.4.2 Event Filtering

**Blank Control Logic:** The DCAEVT1/2 and DCBEVT1/2 events can be filtered using 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 can be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blank control 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. Blank control logic is used to define a blanking window, which ignores all event occurrences on the signal while the window is active. The blanking window is configured in the DCFCTL, DCFOFFSET, and DCFWINDOW registers. The DCFCTL register enables the blanking window and aligns the blanking window to either a CTR = PRD pulse or a CTR = 0 pulse or both CTR = PRD and CTR = 0 as specified by DCFCTL[PULSESEL]. DCFCTL[SRCSEL] selects the DCxEV<sub>Ty</sub> event source for the DCEVTFILT signal. 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. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before.

Figure 13-54 shows the details of the event filtering logic.

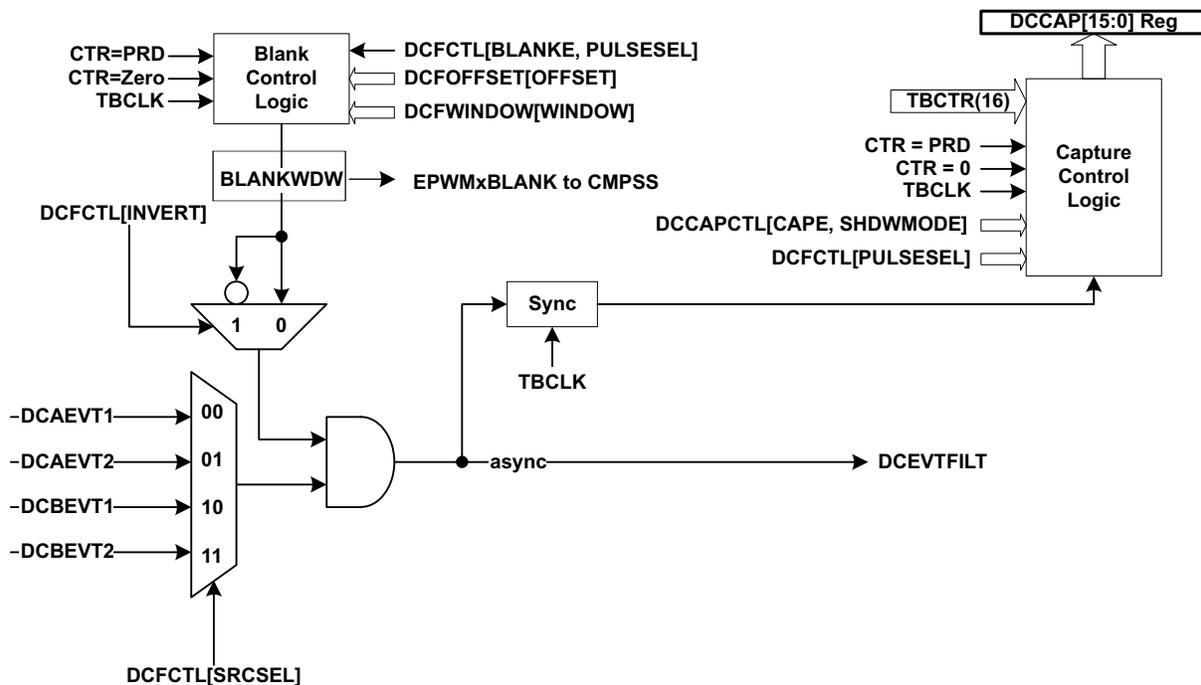


Figure 13-54. Event Filtering

**Capture Control Logic:** The event filtering can also capture the TBCTR value of the selected DCxEV<sub>Ty</sub> event as configured in the DCCAPCTL register. When capture control logic is enabled, the selected DCxEV<sub>Ty</sub> event triggers capture of the TBCTR to the active register. The CPU reads directly from the active register unless shadow mode is enabled by DCCAPCTL[SHDWMODE]. When shadow mode is enabled, the active register information is copied to shadow register on the event specified by DCFCTL[PULSESEL], and the CPU reads from the shadow register. After the selected DCxEV<sub>Ty</sub> event, no further capture events occur until the event specified by DCCAPCTL[CAPMODE]. The CAPMODE can be configured two ways: (1) no further capture events occur until the event defined by DCFCTL[PULSESEL] or (2) no further capture events occur until the compare-event flag at DCCAPCTL[CAPSTS] is cleared by DCCAPCTL[CAPCLR].

#### Note

You must configure the ePWM blanking window appropriately so that the Trip Input stays valid for at least 3 ePWM cycles after the blanking window has expired.

Figure 13-55 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.

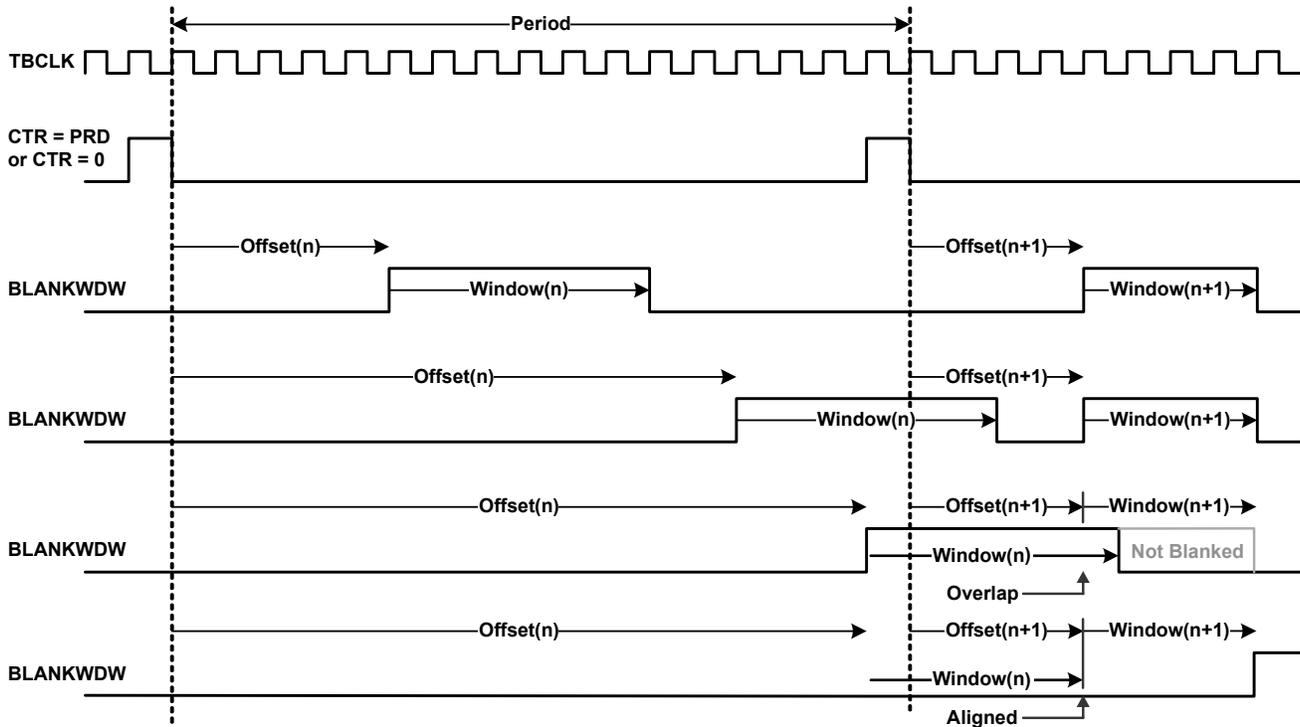


Figure 13-55. Blanking Window Timing Diagram

### BLANKPULSEMIX Signals

The DCFCTL MUX (available for Blank Control Logic and Capture Control Logic) has new options that allows the mux to select the BLANKPULSEMIX signal. The BLANKPULSEMIX signal is used, if the signal is selected by DCFCTL[PULSESEL]

### 13.11.4.3 Valley Switching

Event filtering depicts the valley switching function along with the event filtering logic described in Event Filtering. This function can be used to achieve programmable valley switching without any additional external circuitry. This module provides an on-chip hardware mechanism that can:

- Capture the oscillation period
- Accurately delay the PWM switching instant
- Allow a programmable number of edges before the delay takes effect
- Provide multiple choices of triggers and events
- Allow easy adaptability for optimum performance under changing system/operating conditions

The DCxEV<sub>Ty</sub> signal needs further processing to support valley switching. Here is a brief description of how valley switching function is enabled:

1. Select one of the DCxEV<sub>Ty</sub> events as input to the valley switching block (DCFCTL[*SRCSEL*]) with an option to add the blanking window (Blank Control Logic). This is where the comparator output (or external input) above is selected as an input to the valley switching block.
2. Configure the edge filter to capture 'n' rising, falling or both edges through the edge selection logic (DCFCTL[*EDGEMODE*, *EDGECOUNT*]).
3. Select the correct event to reset and restart the edge filter (VCAPCTL[*TRIGSEL*]). Edge capturing event is triggered or armed by this selected edge.
4. Enable valley capture logic (VCAPCTL[*VCAPE*]).
5. Select the start edge that indicates the start of capture for oscillation period measurement (VCNTCFG[*STARTEDGE*]). This is where the 16-bit counter starts counting.
6. Select the stop edge (VCNTCFG[*STOPEDGE*]) that indicates the edge at which the 16-bit counter stops counting. The captured counter value (CNTVAL) provides oscillation period information.
  - The STOPEDGE value must always be greater than STARTEDGE value.
7. Configure and apply the captured delay (CNTVAL) to the edge filtered DCxEV<sub>Ty</sub> signal. The CNTVAL value can be applied as is or applied in conjunction with a software programmed value (useful for offset adjustment) (SWVDELVAL) or only a fraction of the delay can be applied with or without SWVDELVAL. This is useful to correctly apply a delay corresponding to the valley point. (VCAPCTL[*VDELAYDIV*])
8. Configure VCAPCTL[*EDGEFILTDLYSEL*] to apply hardware delay based on the captured value above.

Once the counter is stopped, counter value is copied into CNTVAL register and counter is reset to zero. No further captures are done until the logic is triggered again by occurrence of event selected by VCAPCTL[*TRIGSEL*]. In this implementation, the software trigger is used as the source for VCAPCTL[*TRIGSEL*]. Upon occurrence of the trigger event, irrespective of the current status of the counter, the counter is reset and starts counting from zero upon occurrence of the STARTEDGE. Similarly, upon occurrence of the trigger event, the edge filter is reset and starts counting from zero upon occurrence of the STARTEDGE.

Output from the valley switching block (DCEVTFILT) is then used to synchronize the PWM time-base. The process is shown in [Figure 13-56](#).

---

#### Note

A specific application example showcasing the usage of valley switching hardware and software is available in C2000Ware.

---

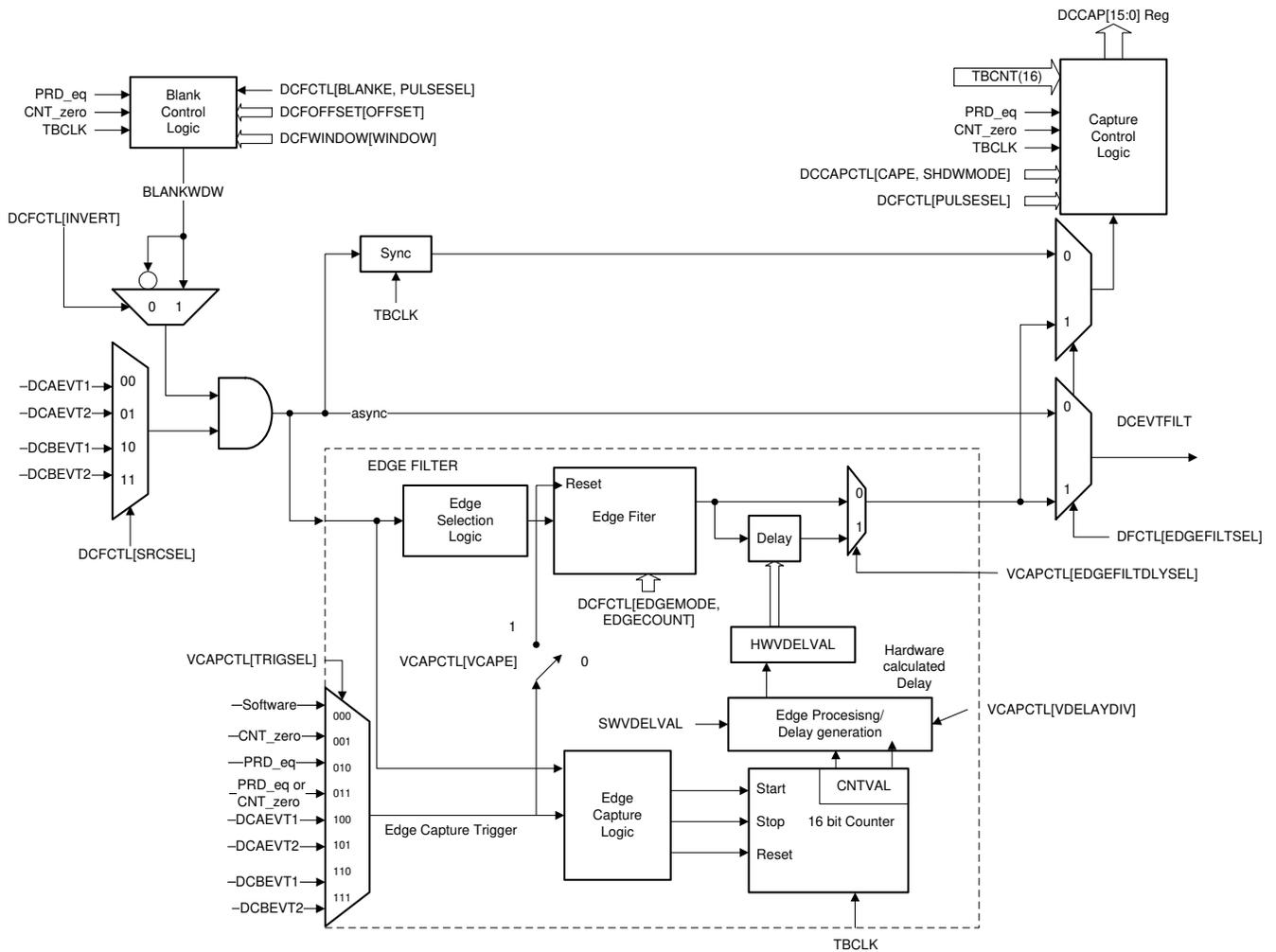


Figure 13-56. Valley Switching

### 13.12 ePWM Crossbar (X-BAR)

Figure 13-57 shows the architecture of the ePWM Crossbar (X-BAR). This module enables selection of various trigger sources into any of the eight dedicated EPWM trips inputs, namely the TRIP4, TRIP5, TRIP7, TRIP8, TRIP9, TRIP10, TRIP11, and TRIP12.

**Note**

Refer to the *Crossbar (X-BAR)* chapter for more information on the X-BAR modules, including X-BAR flags.

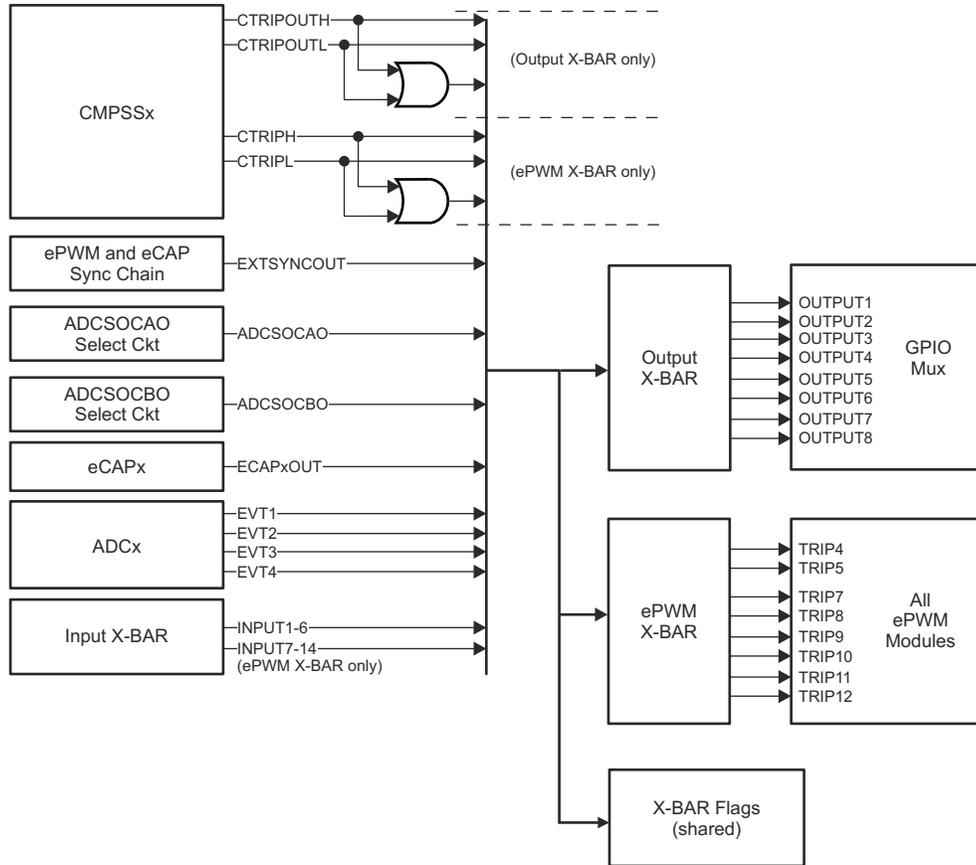


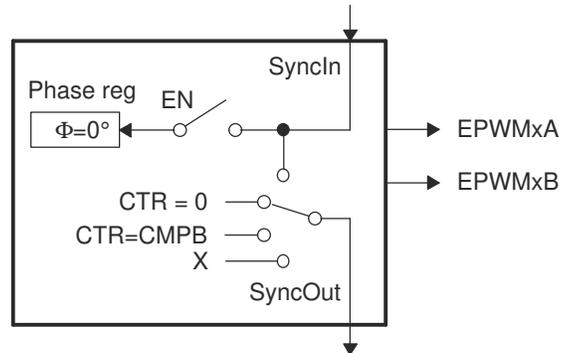
Figure 13-57. ePWM X-BAR

## 13.13 Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

### 13.13.1 Overview of Multiple Modules

Previously in this chapter, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in Figure 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.



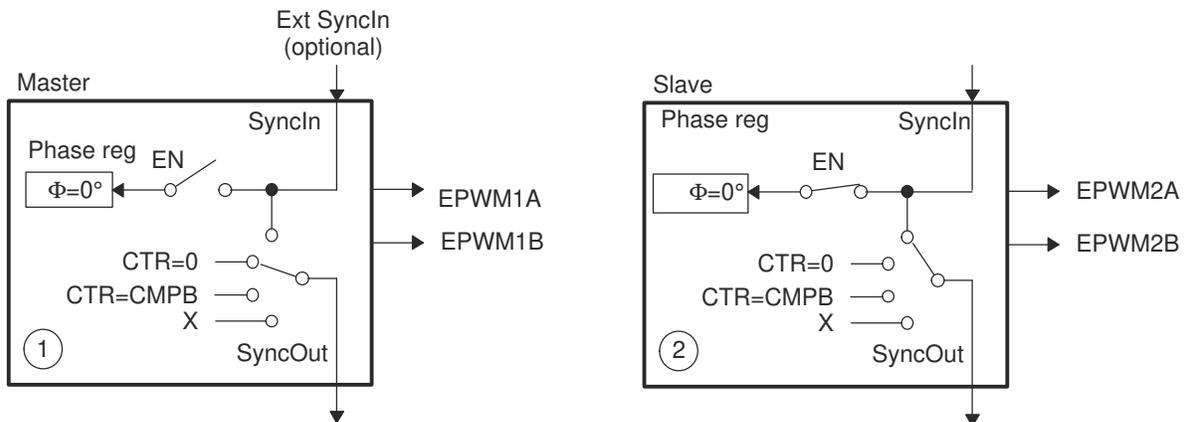
**Figure 13-58. Simplified ePWM Module**

### 13.13.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides no sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides no sync to other modules—SyncOut connected to X (disabled)

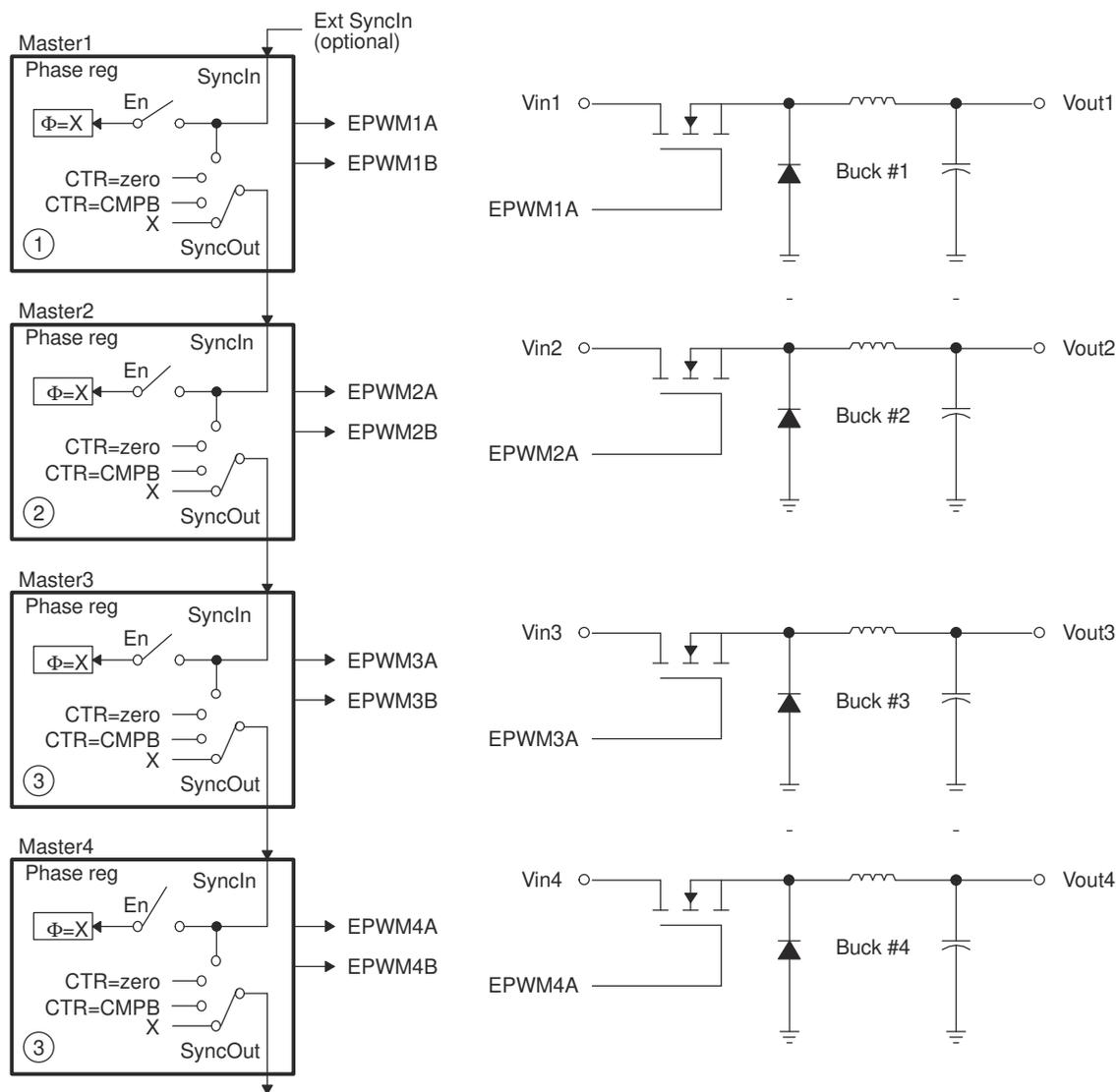
For each choice of SyncOut, a module can also choose to load the counter with a new phase value on a SyncIn strobe input or choose to ignore the value (that is, by the enable switch). Although various combinations are possible, the two most common—Master module and Slave module modes—are shown in [Figure 13-59](#).



**Figure 13-59. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave**

### 13.13.3 Controlling Multiple Buck Converters With Independent Frequencies

One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 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.



A.  $\phi = X$  indicates value in phase register is a "don't care"

**Figure 13-60. Control of Four Buck Stages. Here  $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$**

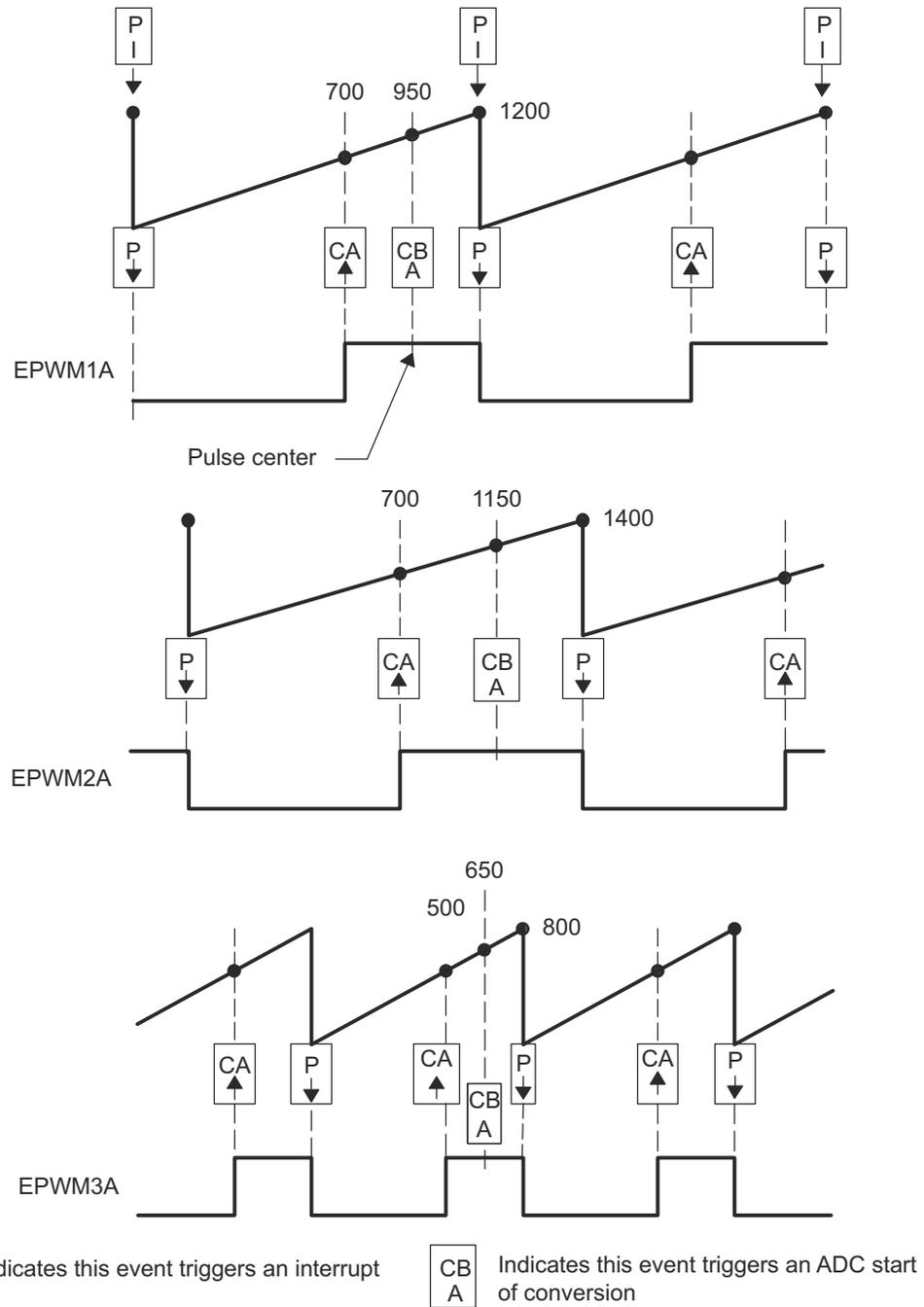
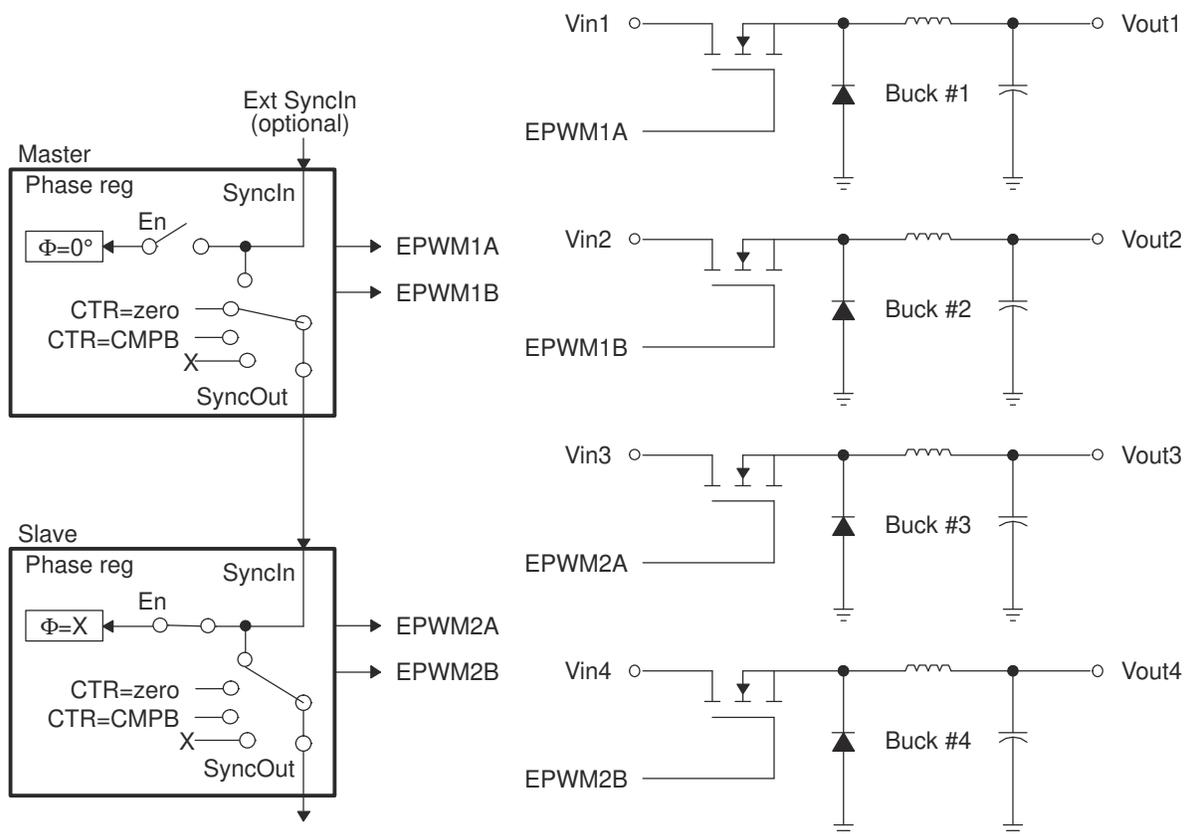


Figure 13-61. Buck Waveforms for Control of Four Buck Stages (Note: Only three bucks shown here)

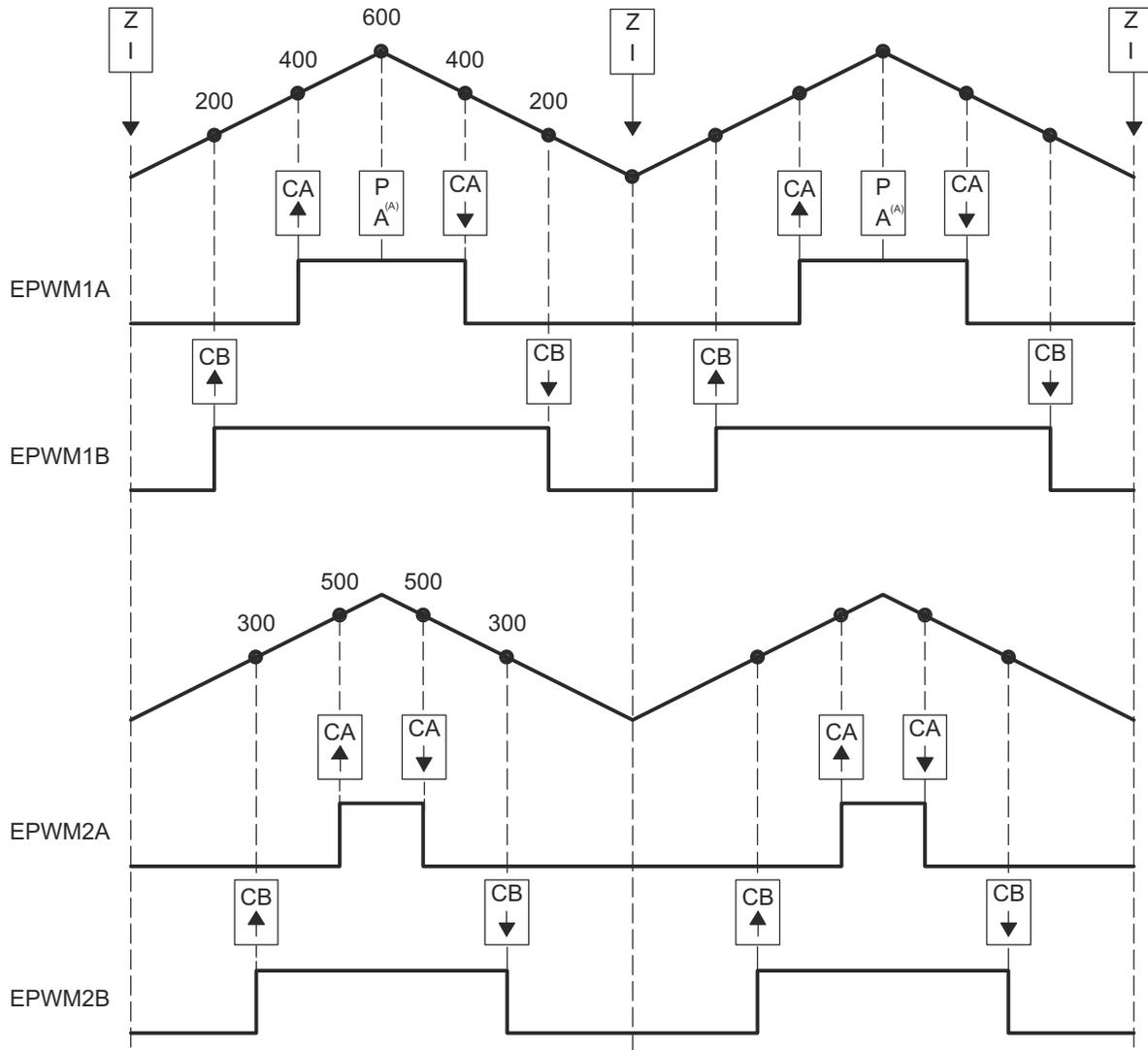
### 13.13.4 Controlling Multiple Buck Converters With Same Frequencies

If synchronization is a requirement, ePWM module 2 is configured as a slave and operates at integer multiple (N) frequencies of module 1. The sync signal from master to slave makes sure these modules remain locked. Figure 13-62 shows such a configuration; Figure 13-63 shows the waveforms generated by the configuration.



A.  $\phi = X$  indicates value in phase register is a "don't care"

**Figure 13-62. Control of Four Buck Stages. (Note:  $F_{P_{WM2}} = N \times F_{P_{WM1}}$ )**



A. Starts ADC conversion.

**Figure 13-63. Buck Waveforms for Control of Four Buck Stages (Note:  $F_{PWM2} = F_{PWM1}$ )**

### 13.13.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 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.

ePWM module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by ePWM module 3 and also, most importantly, to remain in synchronization with master ePWM module 1.

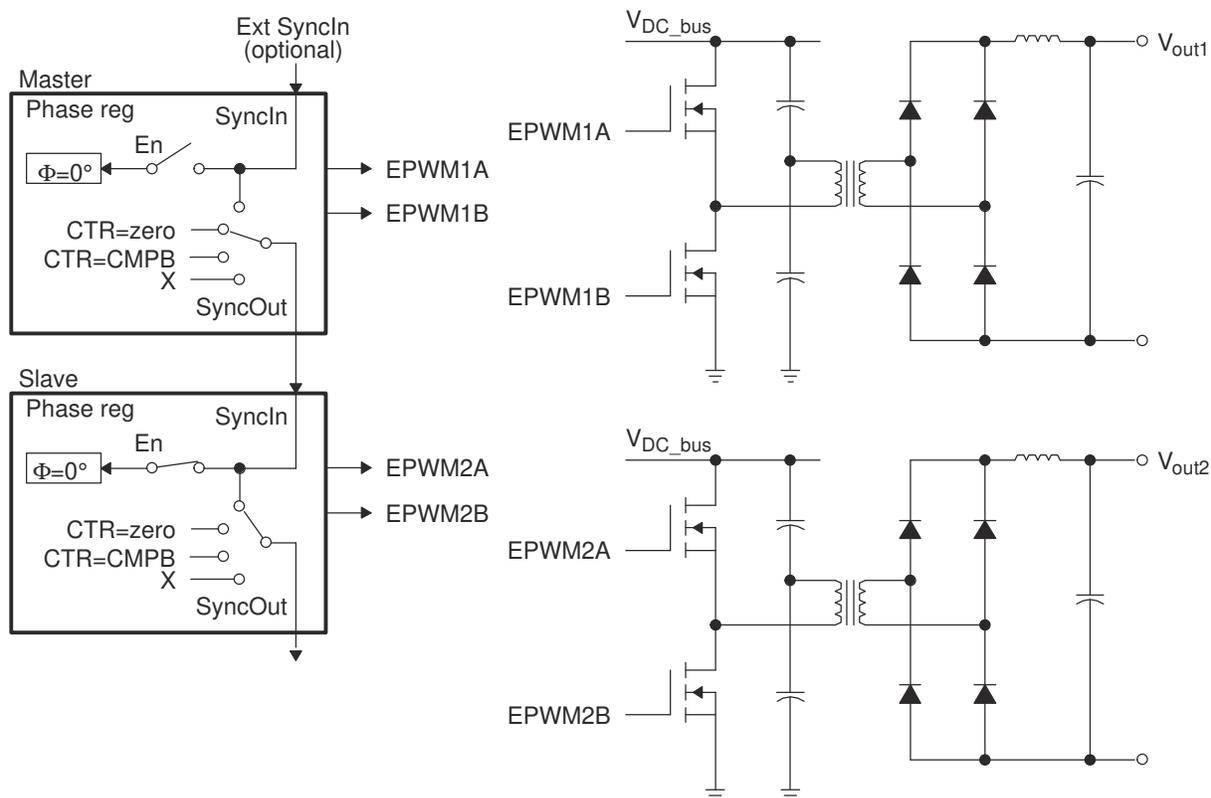


Figure 13-64. Control of Two Half-H Bridge Stages ( $F_{PWM2} = N \times F_{PWM1}$ )

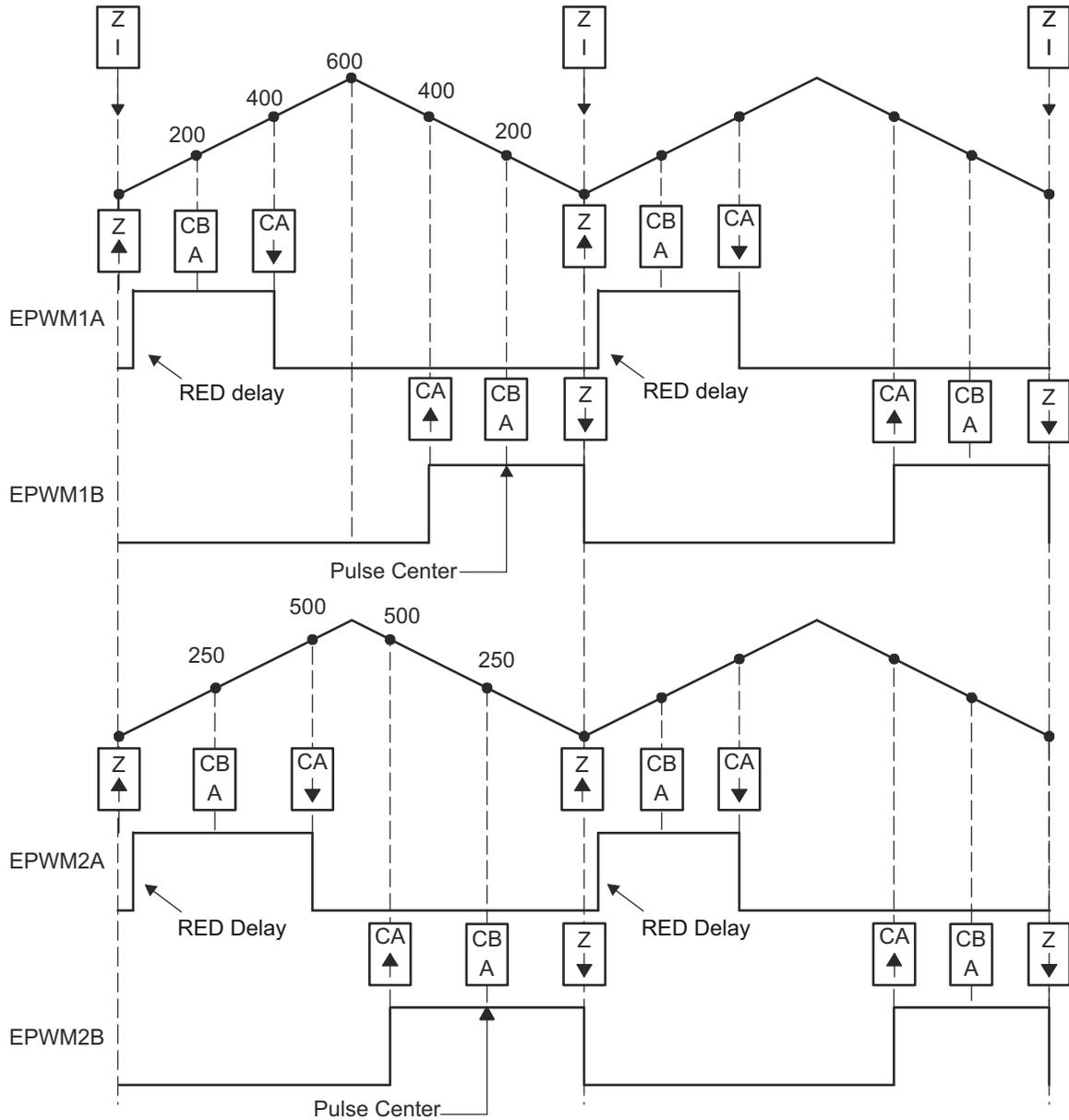
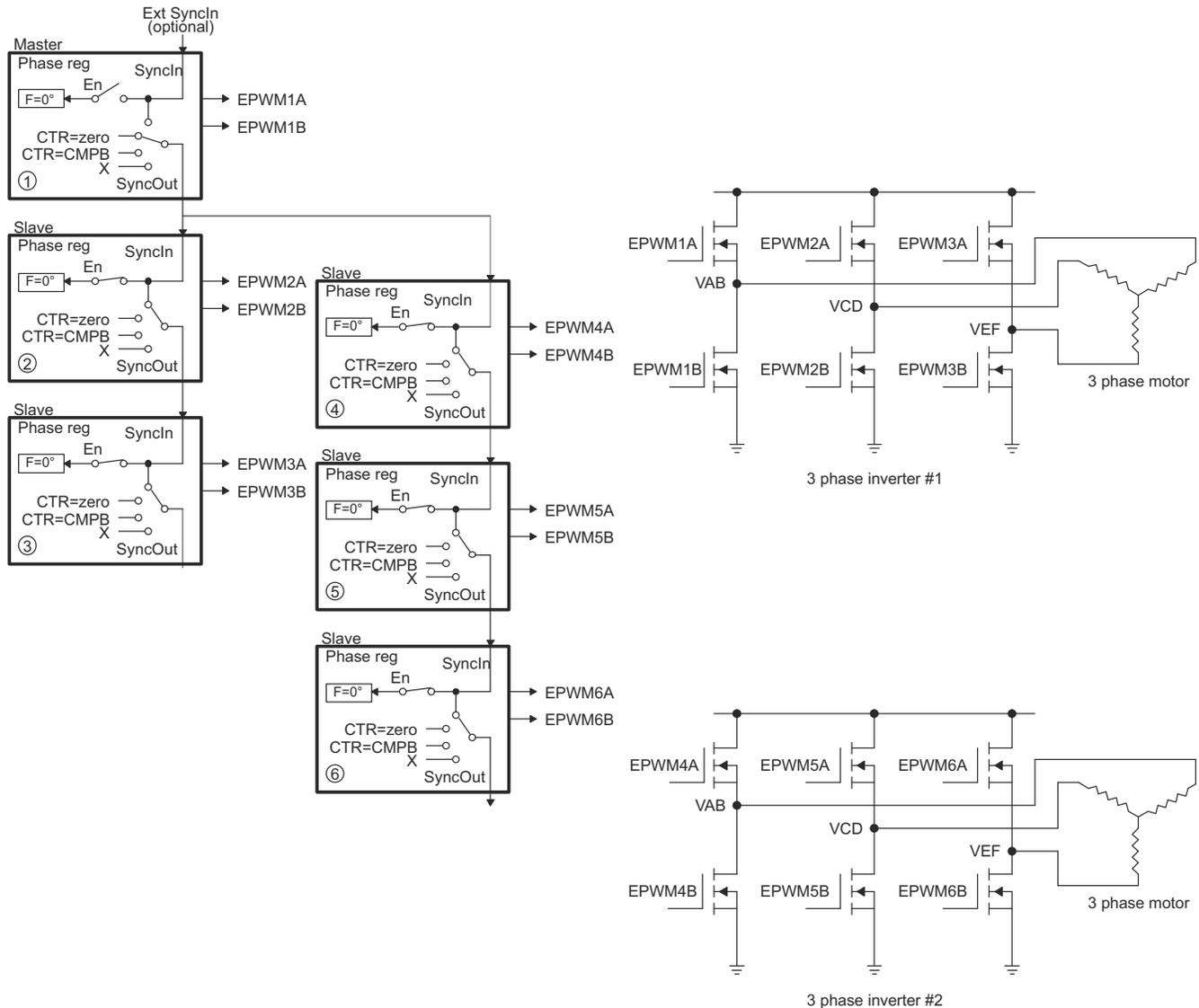


Figure 13-65. Half-H Bridge Waveforms for Control of Two Half-H Bridge Stages (Note: Here  $F_{PWM2} = F_{PWM1}$ )

### 13.13.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase inverter case. In such a case, six switching elements are 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 slaves configuration easily addresses this requirement. Figure 13-66 shows how six PWM modules 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, and 3 (also all equal).



**Figure 13-66. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control**

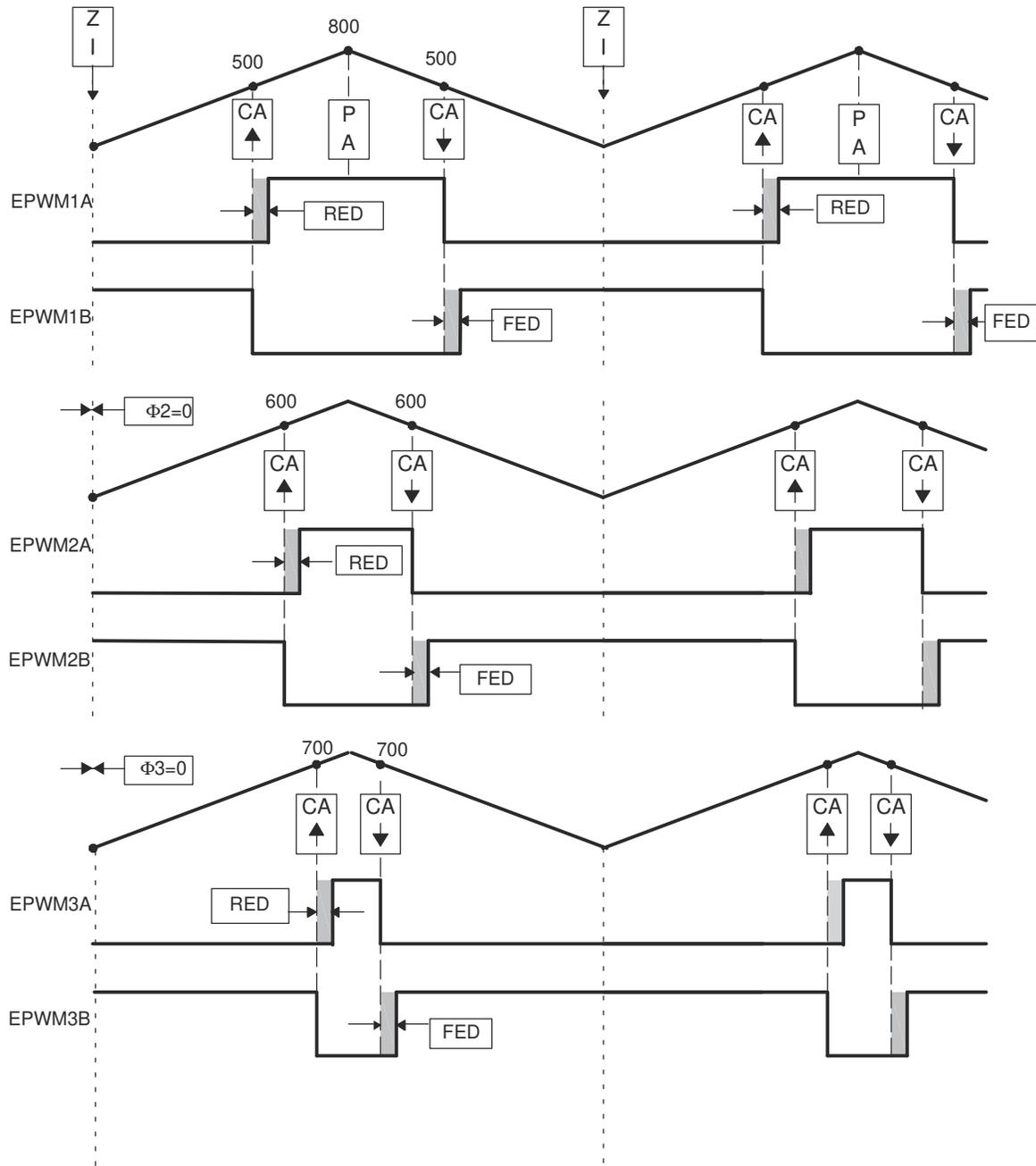
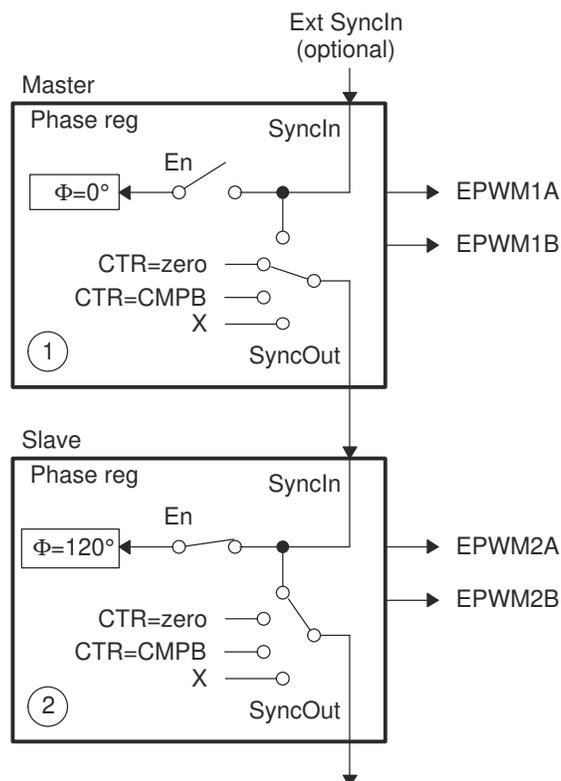


Figure 13-67. 3-Phase Inverter Waveforms for Control of Dual 3-Phase Inverter Stages (Only One Inverter Shown)

### 13.13.7 Practical Applications Using Phase Control Between PWM Modules

So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the time-base submodule section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, [Figure 13-68](#) shows a master and slave module with a phase relationship of  $120^\circ$  (that is, the slave leads the master).



**Figure 13-68. Configuring Two PWM Modules for Phase Control**

[Figure 13-69](#) shows the associated timing waveforms for this configuration. Here,  $TBPRD = 600$  for both master and slave. For the slave,  $TBPHS = 200$  (that is,  $200/600 \times 360^\circ = 120^\circ$ ). Whenever the master generates a SyncIn pulse ( $CTR = PRD$ ), the value of  $TBPHS = 200$  is loaded into the slave TBCTR register so the slave time-base is always leading the master time-base by  $120^\circ$ .

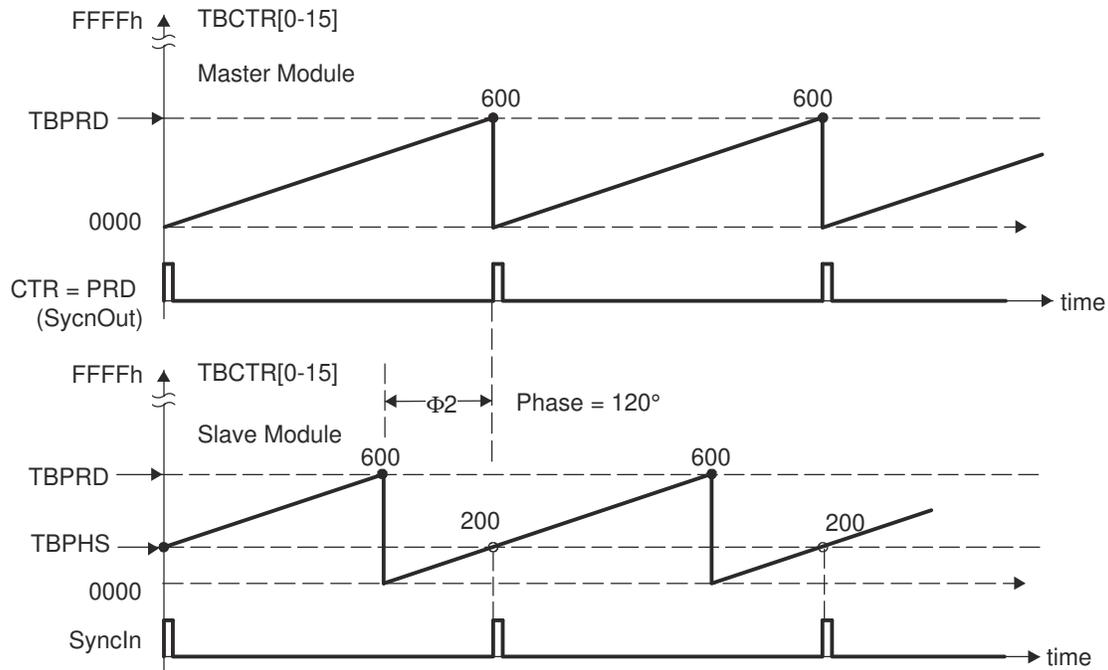


Figure 13-69. Timing Waveforms Associated with Phase Control Between Two Modules

### 13.13.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in [Figure 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 module 1.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

$$TBPHS(N,M) = (TBPRD/N) \times (M-1)$$

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N=3), TBPRD = 600,

TBPHS(3,2) = (600/3) x (2-1) = 200 (that is, Phase value for Slave module 2)

TBPHS(3,3) = 400 (that is, Phase value for Slave module 3)

[Figure 13-71](#) shows the waveforms for the configuration in [Figure 13-70](#).

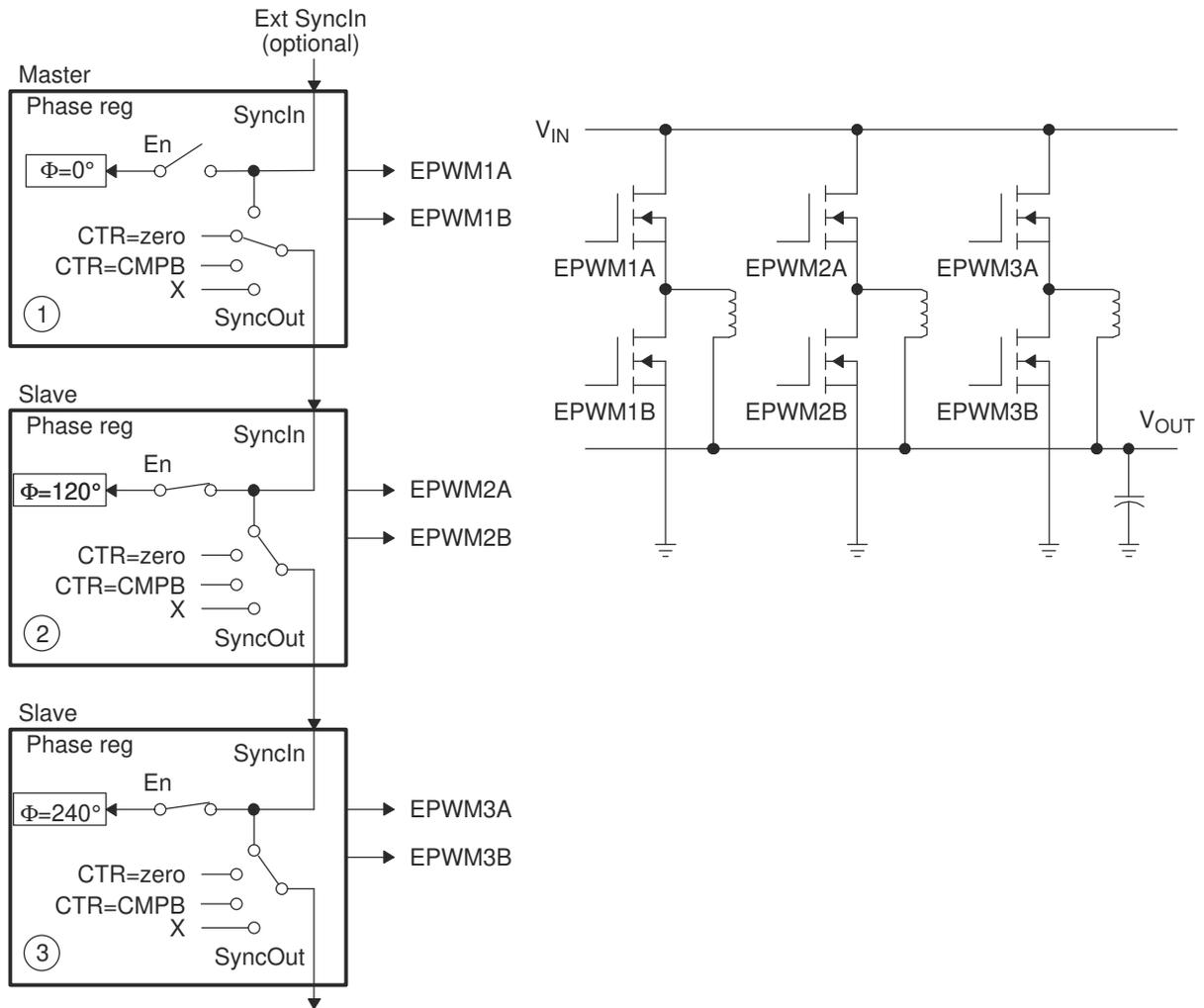


Figure 13-70. Control of 3-Phase Interleaved DC/DC Converter

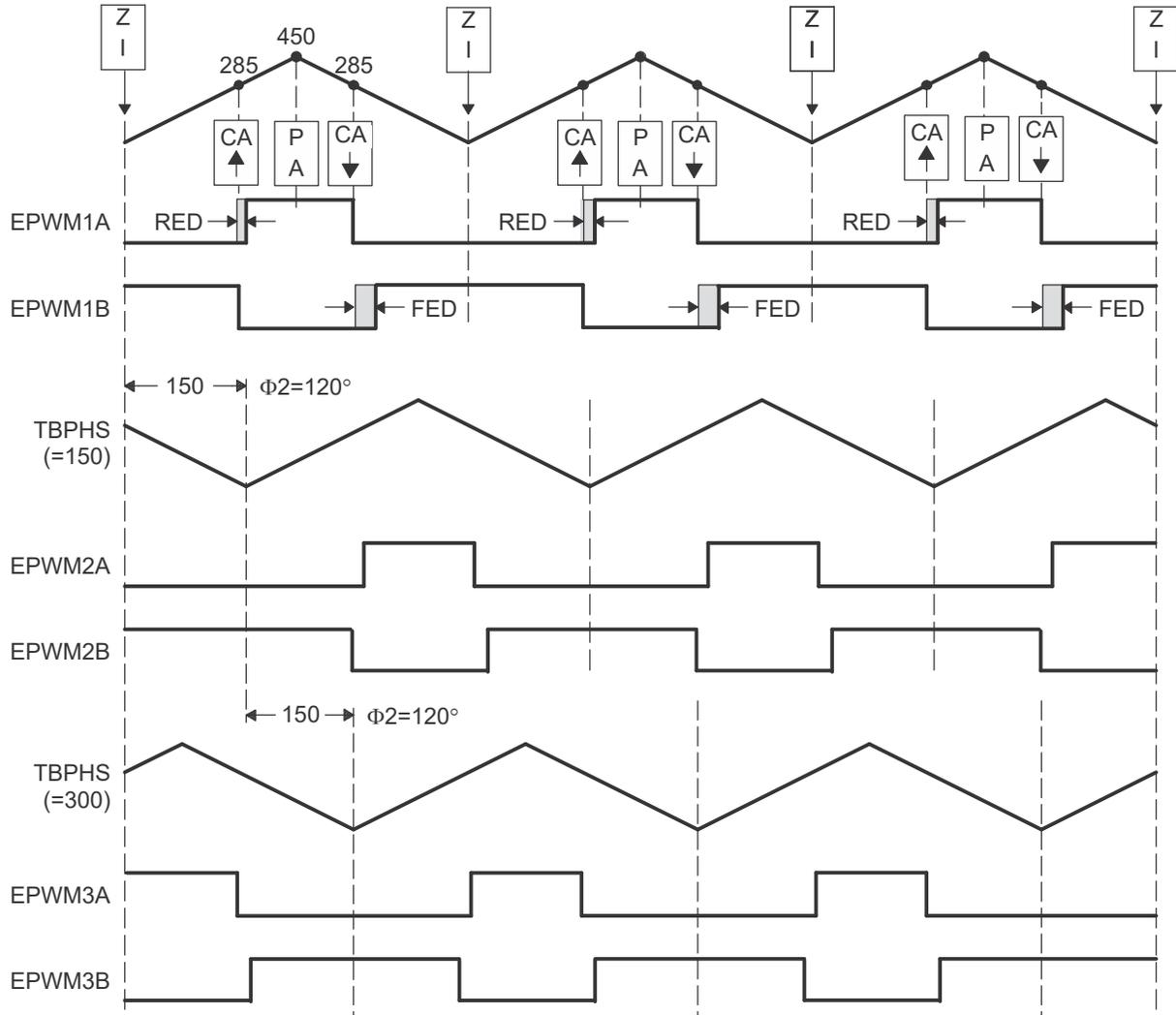
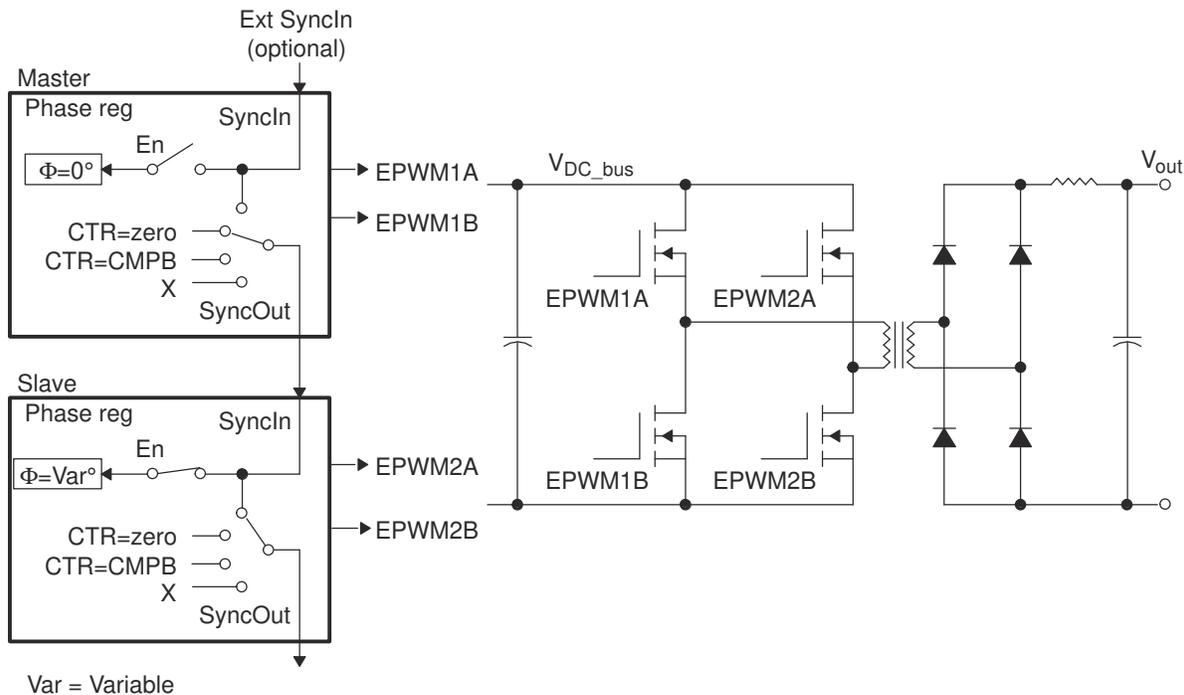


Figure 13-71. 3-Phase Interleaved DC/DC Converter Waveforms for Control of 3-Phase Interleaved DC/DC Converter

### 13.13.9 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 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 and 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 phase register (TBPHS). The master phase register is not used and therefore can be initialized to zero.



**Figure 13-72. Control of Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ )**

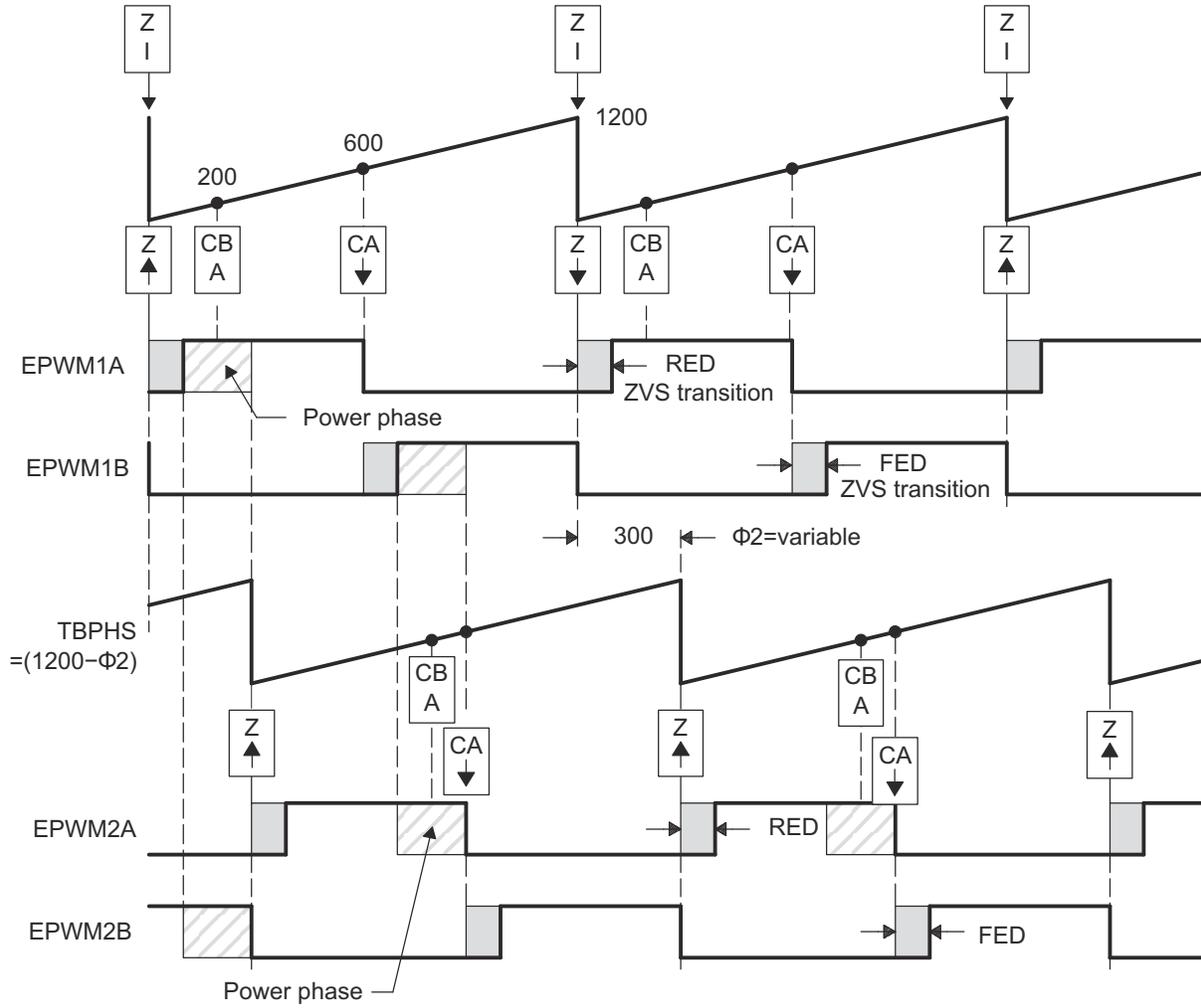


Figure 13-73. ZVS Full-H Bridge Waveforms

### 13.13.10 Controlling a Peak Current Mode Controlled Buck Module

Peak current control techniques offer a number of benefits like automatic over current limiting, fast correction for input voltage variations and reducing magnetic saturation. Figure 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 12-bit DAC can be used to provide a reference peak current at the negative terminal of the comparator. Alternatively, an external reference can 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.

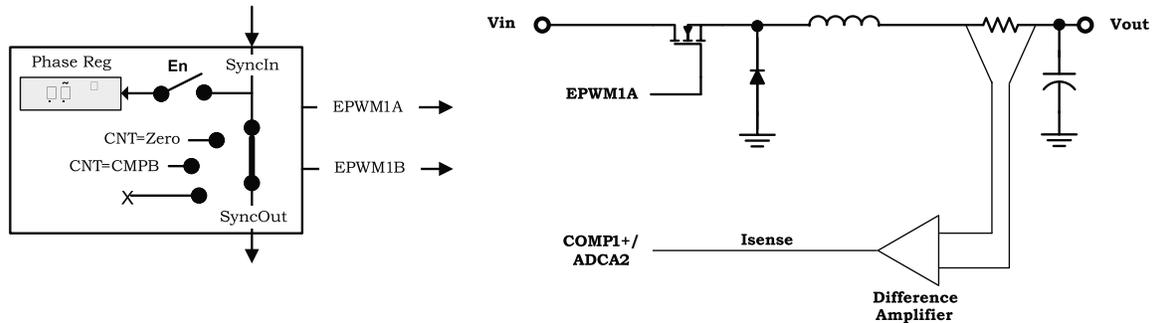


Figure 13-74. Peak Current Mode Control of Buck Converter

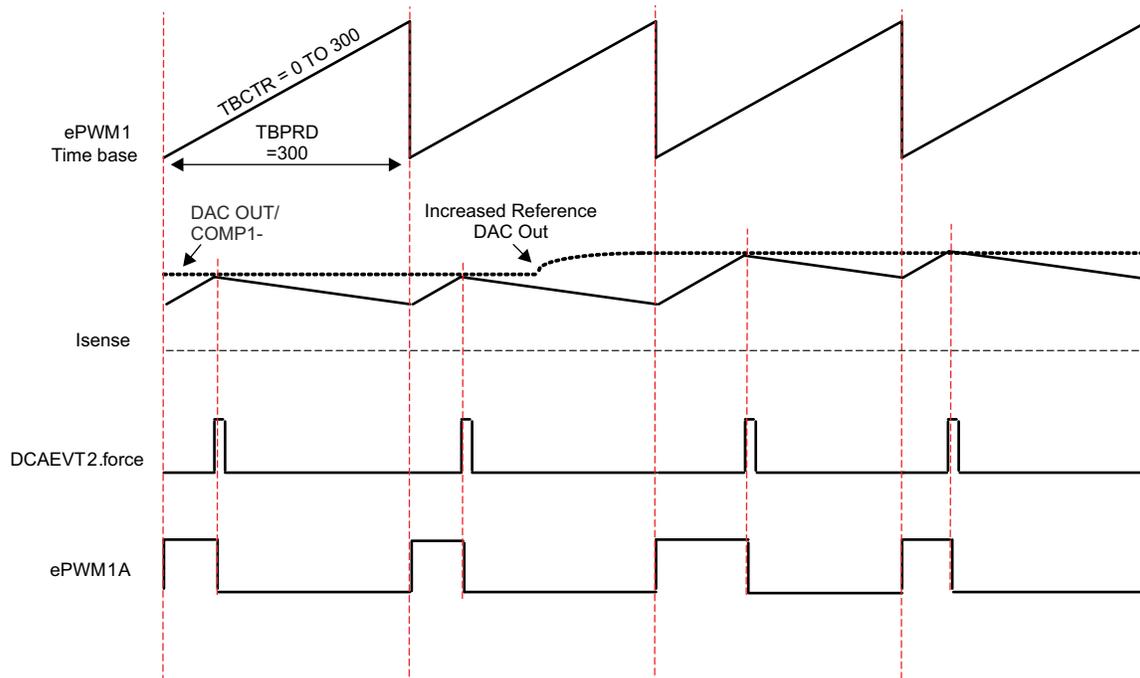
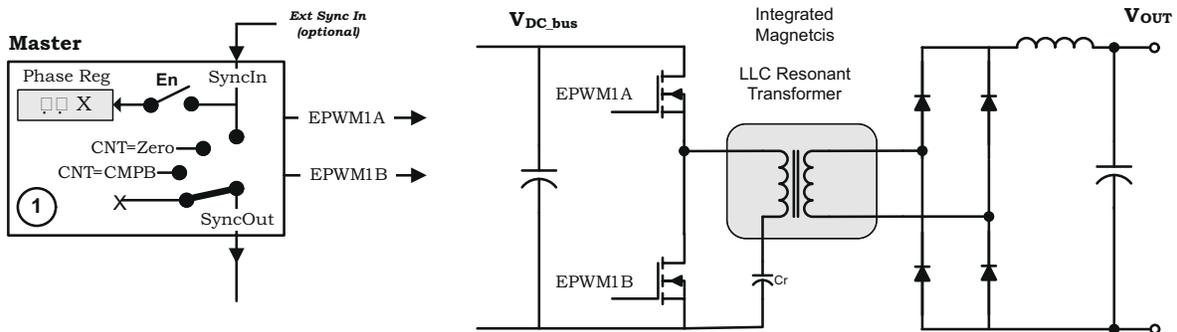


Figure 13-75. Peak Current Mode Control Waveforms for Control of Buck Converter

### 13.13.11 Controlling H-Bridge LLC Resonant Converter

Various topologies of resonant converters are well-known in the field of power electronics for many years. In addition to these, H-bridge LLC resonant converter topology has recently gained popularity in many consumer electronics applications where high efficiency and power density are required. In this example, single channel configuration of ePWM1 is detailed, yet the configuration can easily be extended to multichannel. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead the parameter is frequency. Although the deadband is not controlled and kept constant as 300ns (that is, 30 at 100MHz TBCLK), the user can update the deadband in real time to enhance the efficiency by adjusting enough time delay for soft switching.



NOTE:  $\Theta = X$  indicates value in phase register is 'don't care'

Figure 13-76. Control of Two Resonant Converter Stages

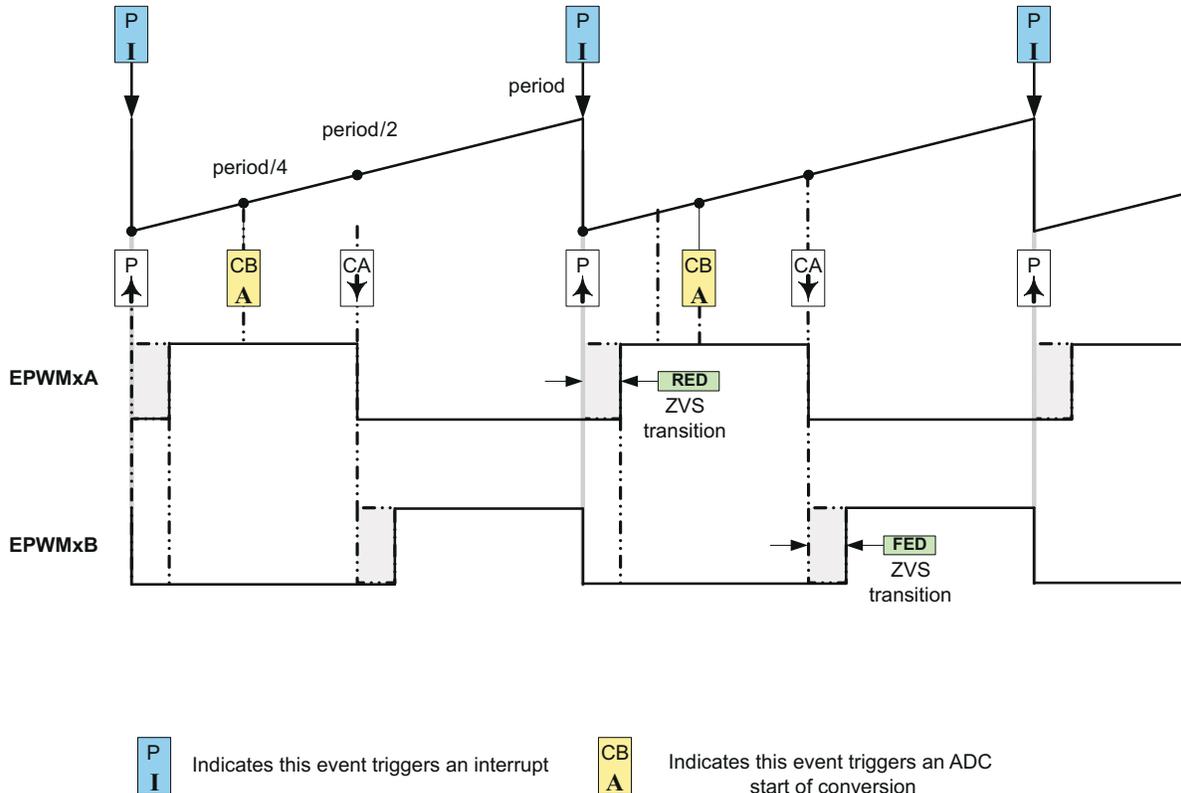


Figure 13-77. H-Bridge LLC Resonant Converter PWM Waveforms

## 13.14 Register Lock Protection

The register lock protection mechanism is added to protect the critical ePWM registers from being corrupted by accidental writes in case of runaway code. The register EPWMLOCK contains the definition of Lock bits (Table 13-14 shows the lock bits and the corresponding registers). This register also has a KEY field; writes to this register succeed only if the KEY field is written with a value of 0xa5a5. Refer to the register descriptions for more details.

**Table 13-14. Lock Bits and Corresponding Registers**

Bit Field	Definition	Registers Locked
HRLOCK	HRPWM Register Set Lock	HRCNFG, HRPWR, HRMSTEP, HRPCTL
GLLOCK	Global Load Register Set Lock	GLDCTL, GLDCFG
TZCFGLOCK	TripZone Register Set Lock	TZSEL, TZDCSEL, TZCTL, TZCTL2, TZCTLDCA, TZCTLDCB, TZEINT
TZCLRLOCK	TripZone Clear Register Set Lock	TZCLR, TZCBCCLR, TZOSTCLR, TZFRC
DCLOCK	Digital Compare Register Set Lock	DCTRIPSEL, DCACTL, DCBCTL, DCFCTL, DCCAPCTL, DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL, DCBLTRIPSEL

### Note

Due to the presence of the KEY field in the same register, only 32-bit writes succeed if the KEY matches. The 16-bit writes to the upper or lower half of this register are ignored.

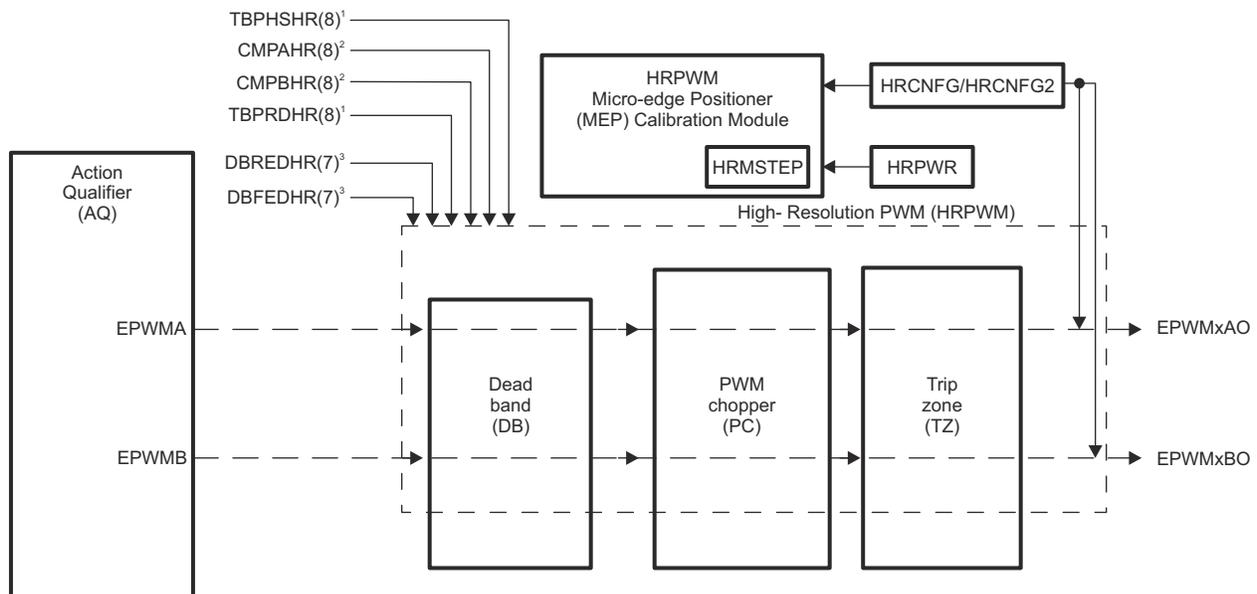
### 13.15 High-Resolution Pulse Width Modulator (HRPWM)

Figure 13-78 shows a block diagram of the HRPWM. This module extends the time resolution capabilities of the conventionally derived digital pulse width modulator (PWM). HRPWM is typically used when PWM resolution falls below approximately 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 and 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 how designed
- Enables high-resolution output swapping on the EPWMxA and EPWMxB output
- Enables high-resolution output on EPWMxB signal output using 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

**Note**

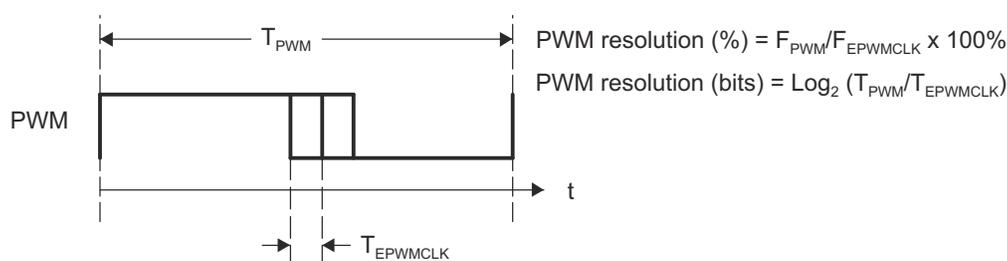
See the device data sheet to determine if your device has an ePWM module with high-resolution period support.



- A. From ePWM Time-base (TB) submodule
- B. From ePWM counter-compare (CC) submodule
- C. From ePWM Deadband (DB) submodule

**Figure 13-78. HRPWM Block Diagram**

The ePWM peripheral is used to perform a function mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 13-79, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.



**Figure 13-79. Resolution Calculations for Conventionally Generated PWM**

If the required PWM operating frequency does not offer sufficient resolution in PWM mode, consider using HRPWM. As an example of improved performance offered by HRPWM, Table 13-15 shows resolution in bits for various PWM frequencies. These values assume a MEP step size of 180ps. See the device data sheet for typical and maximum performance specifications for the MEP.

**Table 13-15. Resolution for PWM and HRPWM**

PWM Frequency (kHz)	Regular Resolution (PWM) 100MHz EPWMCLK		High Resolution (HRPWM)	
	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 can differ, typical low-frequency PWM operation (below 250kHz) does not require HRPWM. HRPWM capability is most useful for high-frequency PWM requirements of power conversion topologies such as:

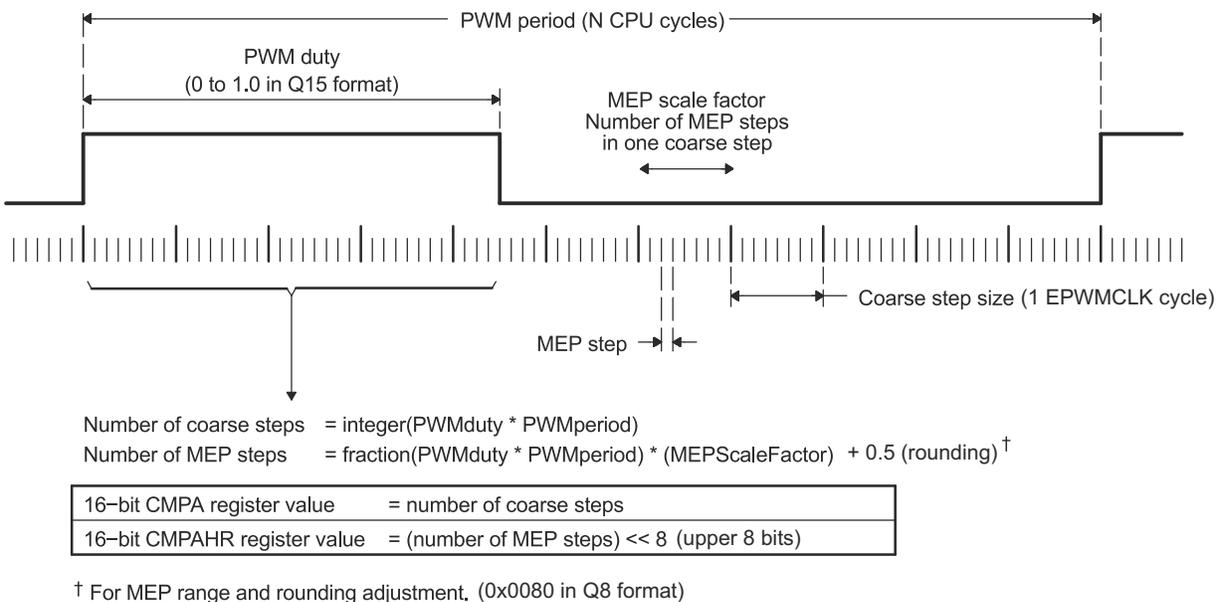
- Single-phase buck, boost, and flyback
- Multiphase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

### 13.15.1 Operational Description of HRPWM

The HRPWM is based on micro-edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150ps. See the device 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 as designed under all operating conditions. Details on software diagnostics and functions are in [Section 13.15.1.7](#).

[Figure 13-80](#) shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled using an 8-bit field in the Compare A extension register (CMPAHR). The same operating logic applies to CMPBHR as well.

To generate an HRPWM waveform, configure the ePWM registers 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 13.15.1.8](#).

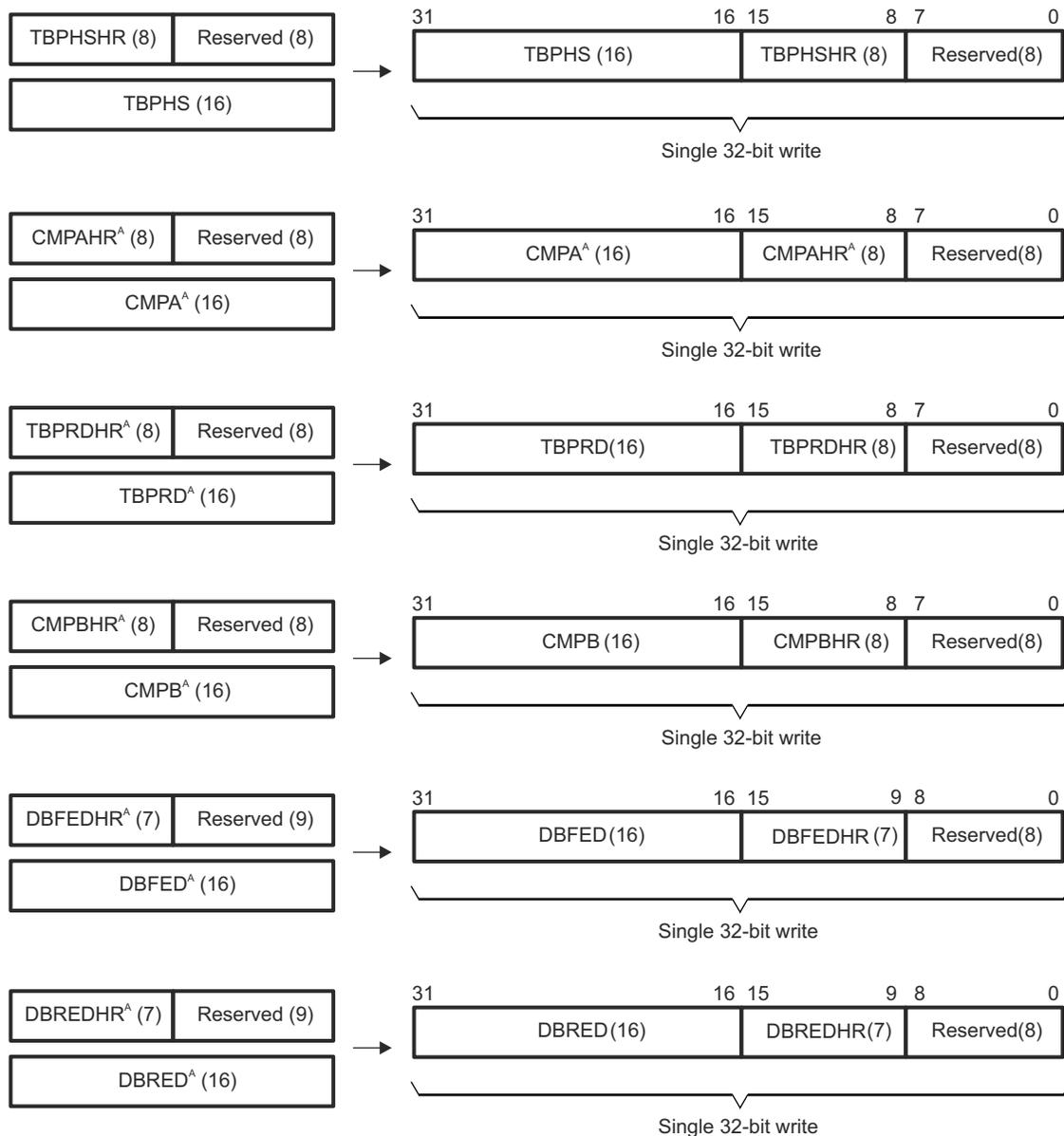


**Figure 13-80. Operating Logic Using MEP**

#### 13.15.1.1 Controlling the HRPWM Capabilities

The MEP of the HRPWM is controlled by six extension registers. These HRPWM registers are concatenated with the 16-bit TBPHS, TBPRD, CMPA, CMPBM, DBREDM, and DBFEDM registers used to control PWM operation.

- TBPHSHR - Time Base Phase High Resolution Register
- CMPAHR - Counter Compare A High Resolution Register; CMPAHR is for use with the AQ output of Channel A, and is not related to CMPA
- TBPRDHR - Time Base Period High Resolution Register. (available on some devices)
- CMPBHR - Counter Compare B High Resolution Register; CMPBHR is for use with the AQ output of Channel B, and is not related to CMPB
- DBREDHR - Dead-band Generator Rising Edge Delay High Resolution Register
- DBFEDHR - Dead-band Generator Falling Edge Delay High Resolution Register



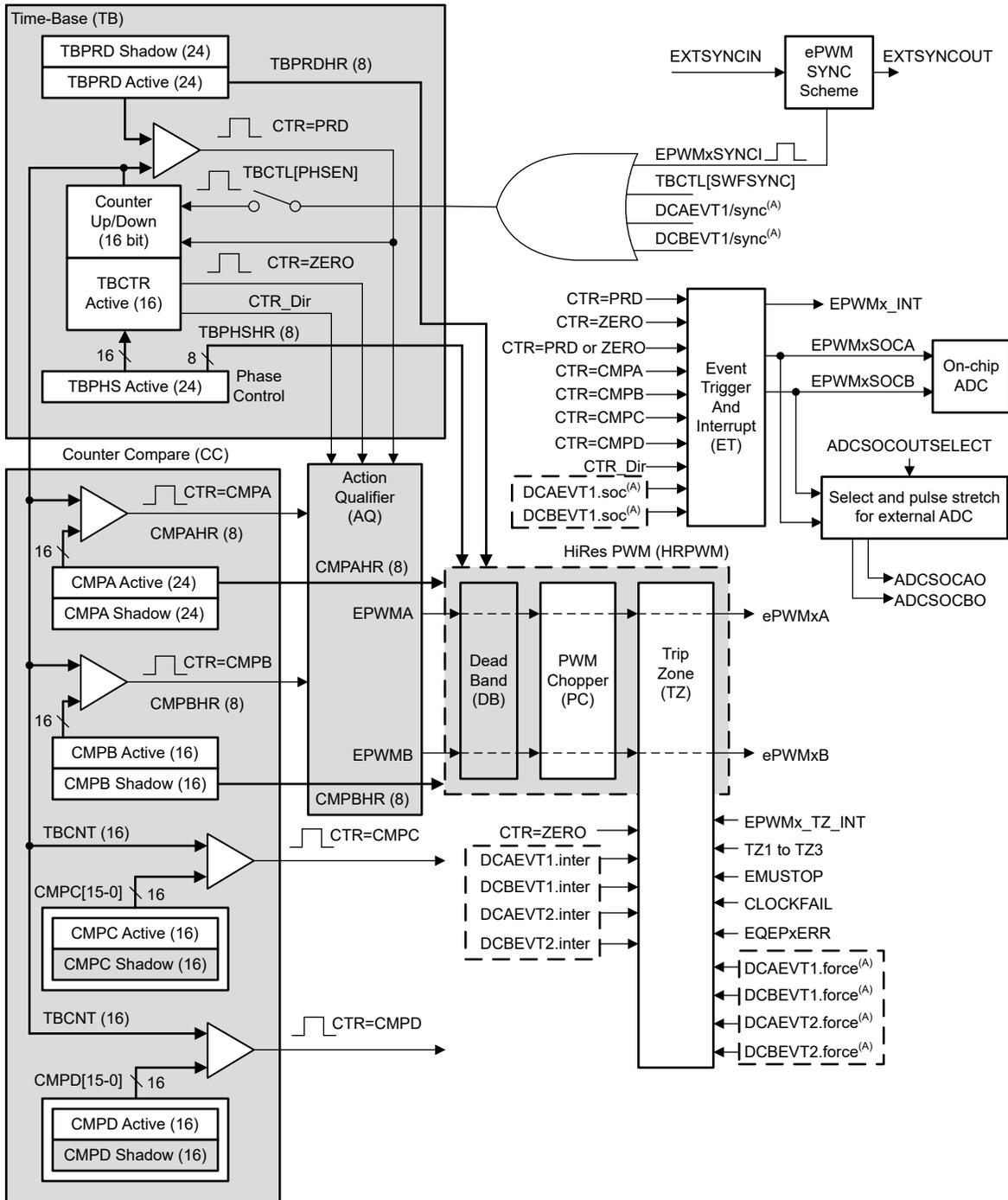
A. Dependent upon your device, these registers can be mirrored and can be written to at two different memory locations.

**Figure 13-81. HRPWM Extension Registers and Memory Configuration**

**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 and B PWM signal path. HRPWM support on the Dead band signal path is available by properly configuring the HRCNFG2 register. Figure 13-82 shows how the HRPWM interfaces with the 8-bit extension registers.

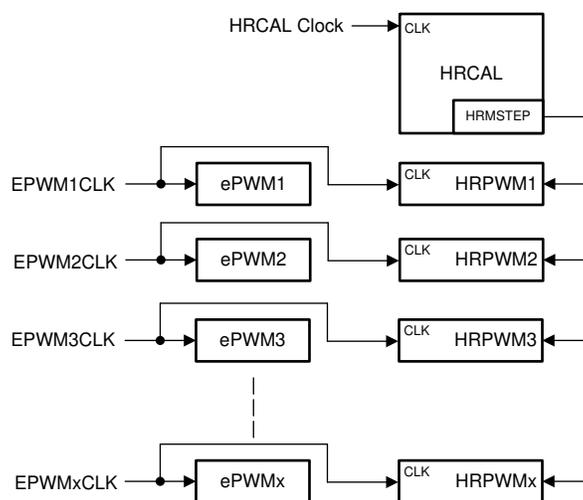


A. These events are generated by the ePWM Digital Compare (DC) submodule based on the levels of the TRIPIN inputs.

**Figure 13-82. HRPWM System Interface**

### 13.15.1.2 HRPWM Source Clock

Each HRPWM module is clocked from the respective EPWMxCLK. HRCAL has a separate clock. For example, HRPWM1 is sourced from EPWM1CLK while HRPWM2 is clocked from the EPWM2CLK. Figure 13-83 shows the HRCAL and HRPWM modules are sourced from their respective ePWM clock source.



**Figure 13-83. HRPWM and HRCAL Source Clock**

### 13.15.1.3 Configuring the HRPWM

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register in that particular ePWM module's register space. This register provides the following configuration options:

- Edge Mode** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE) or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control (CMPA or CMPB high-resolution control), while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge (TBPHS or TBPRD high-resolution control).
- Control Mode** The MEP is programmed to be controlled either from the CMPAHR/CMPBHR register in case of duty cycle control or the TBPHSHR register (phase control). RE or FE control mode can be used with the CMPAHR or CMPBHR register. BE control mode can be used with the TBPHSHR register. When the MEP is controlled from the TBPRDHR register (period control), the duty cycle and phase can also be controlled using 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 can be chosen to be the same as the regular load option for the CMPA/CMPB register. If TBPHSHR is used, then this option has no effect.
- High-Resolution B Signal Control** The B signal path of an ePWM channel can generate a high-resolution output by outputting an inverted version of the high-resolution ePWMxA signal on the ePWMxB pin. A Type 2 or Type 4 HRPWM module can also enable high-resolution features on the B signal path independently of the A signal path as well.
- Swap ePWMxA and ePWMxB Outputs** This mode enables the swapping of the high-resolution A and B outputs. The mode selection allows either A and B Outputs Unchanged or A Output Comes Out On B and B Output Comes Out On A.

**Auto-  
conversion  
Mode**

This mode is used in conjunction with the scale factor optimization (SFO) software only. For a type 4 HRPWM module, below is a description of the Auto-conversion Mode taking CMPAHR as an example. If auto-conversion is enabled,  $CMPAHR = \text{fraction}(PWMduty * PWMperiod) \ll 8$ . The scale factor optimization software calculates the MEP scale factor in the background code and automatically updates the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module then uses the values in the HRMSTEP and CMPAHR registers to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and moves the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, the CMPAHR register behaves like a type 0 HRPWM module and  $CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) \ll 8$ . All calculations need to be performed by your 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**

If the HRPWM module is configured in UP-DOWN counter mode, the shadow mode for the HRPWM registers must be set to load on both ZERO AND PERIOD. New values from the user are loaded to the shadow registers only at CTR=ZERO, but the shadow mode of for the registers must be set to both ZERO AND PERIOD. The CTR=PRD event is used for specific internal logic inside the HRPWM module.

Auto-conversion Mode performs the calculation for CMPBHR, DBREDHR, and DBFEDHR. The scale factor optimization software calculates the MEP scale factor in the background code and automatically updates the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module then uses 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 moves the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, CMPBHR behaves the same as CMPAHR.  $CMPBHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor} + 0.5) \ll 8$ .

You are expected to disable protection for both ePWM1 and HRPWM when the application requires access to either of these modules.

---

### 13.15.1.4 Configuring High-Resolution in Deadband Rising-Edge and Falling-Edge Delay

Once the ePWM has been configured to provide conventional PWM of a given frequency, polarity, and dead band 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 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 the active register for DBRED and DBFED in high-resolution mode. Select the pulse to match the selection in the ePWM DBCTL[LOADREDMODE] and DBCTL[LOADFEDMODE] bits.

### 13.15.1.5 Principle of Operation

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps (see the device data sheet for typical MEP step size). The MEP works with the TBM and CCM registers to be certain that time steps are applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies, and other operating conditions. Table 13-16 shows the typical range of operating frequencies supported by the HRPWM.

**Table 13-16. Relationship Between MEP Steps, PWM Frequency, and Resolution**

System (MHz)	MEP Steps Per EPWMCLK <sup>(1) (2) (3)</sup>	PWM Minimum (Hz) <sup>(4)</sup>	PWM Maximum (MHz)	Resolution at Maximum (Bits) <sup>(5)</sup>
60.0	93	916	3.00	10.9
70.0	79	1068	3.50	10.6
80.0	69	1221	4.00	10.4
90.0	62	1373	4.50	10.3
100.0	56	1526	5.00	10.1

- (1) TBCLK = EPWMCLK.  
 (2) Table data based on a MEP time resolution of 180ps (this is an example value. See the device data sheet for MEP limits)  
 (3) MEP steps applied =  $T_{EPWMCLK}/180ps$  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.

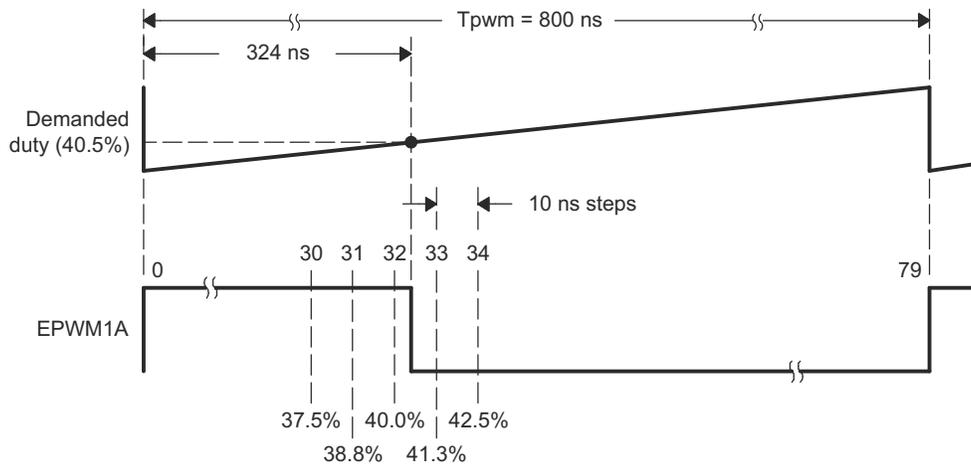
**13.15.1.5.1 Edge Positioning**

**Note**

The following example is presented using the [CMPA:CMPAHR] register combination. The theory of operation and equations are the same, if intending to use the [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 1.25MHz. In conventional PWM generation with a system clock of 100MHz, the duty cycle choices are in the vicinity of 40.5%. As shown in Figure 13-84, 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 320ns instead of the desired 324ns. This data is shown in Table 13-17.

By utilizing the MEP, an edge position much closer to the desired point of 324ns can be achieved. Table 13-17 shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) positions the edge at 323.96ns, resulting in almost zero error. In this example, assume that the MEP has a step resolution of 180ps.



**Figure 13-84. Required PWM Waveform for a Requested Duty = 40.5%**

**Table 13-17. CMPA versus Duty (left), and [CMPA:CMPAHR] versus Duty (right)**

CMPA (count) <sup>(1) (2) (3)</sup>	Duty (%)	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	35.0	280	32	18	<b>40.405</b>	323.24
29	36.3	290	32	19	<b>40.428</b>	323.42
30	37.5	300	32	20	<b>40.450</b>	323.60
31	38.8	310	32	21	<b>40.473</b>	323.78
32	40.0	320	32	22	<b>40.495</b>	323.96
33	41.3	330	32	23	<b>40.518</b>	324.14
34	42.5	340	32	24	<b>40.540</b>	324.32
			32	25	<b>40.563</b>	324.50
Required			32	26	<b>40.585</b>	324.68
32.40	<b>40.5</b>	324	32	27	<b>40.608</b>	324.86

(1) Assumed MEP step size for the above example = 180ps. See the device-specific data sheet for typical and maximum MEP values.  
 (2) TBCLK = 100MHz, 10ns  
 (3) For a PWM Period register value of 80 counts, PWM Period = 80 × 10ns = 800ns, PWM frequency = 1/800ns = 1.25MHz

### 13.15.1.5.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard CMPA and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in nanoseconds (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	= 10ns (100MHz)
PWM frequency	= 1.25MHz (1/800ns)
Required PWM duty cycle, <b>PWMDuty</b>	= 0.405 (40.5%)
PWM period in terms of coarse steps, <b>PWMPeriod</b> (800ns/10ns)	= 80
Number of MEP steps per coarse step at 180ps (10ns/180ps), <b>MEP_ScaleFactor</b>	= 55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	= 0.5 (0080h in Q8 format)

#### Step 1: Percentage Integer Duty value conversion for CMPA register

CMPA register value	= $\text{int}(\text{PWMDuty} * \text{PWMPeriod})$ ; int means integer part
	= $\text{int}(0.405 * 80)$
	= $\text{int}(32.4)$
CMPA register value	= 32 (20h)

#### Step 2: Fractional value conversion for CMPAHR register

CMPAHR	= $(\text{frac}(\text{PWMDuty} * \text{PWMPeriod}) * \text{MEP\_ScaleFactor} + 0.5) \ll 8$ ; frac means fractional part
	= $(\text{frac}(32.4) * 55 + 0.5) \ll 8$ ; Shifting is to move the value to the high byte of CMPAHR.
	= $(0.4 * 55 + 0.5) \ll 8$
	= $(22 + 0.5) \ll 8$
	= $22.5 * 256$ ; Shifting left by 8 is the same as multiplying by 256.
	= 5760 (1680h)
CMPAHR	= 1680h CMPAHR value = 1600h (lower 8 bits are ignored by hardware).

---

### Note

If the AUTOCONV bit (HRCNFG.6) is set and the MEP\_ScaleFactor is in the HRMSTEP register, then CMPAHR / CMPBHR register value =  $\text{frac}(\text{PWMDuty} * \text{PWMperiod} << 8)$ . 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.

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 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 13.15.1.8](#). The actual implementation takes advantage of the 32-bit CPU architecture and is somewhat different from the steps shown in [Section 13.15.1.5.2](#).

For time-critical control loops where every cycle counts, the assembly version is recommended. This is a cycle optimized function (11 EPWMCLK cycles) that takes a Q15 duty value as input and writes a single [CMPA:CMPAHR] value.

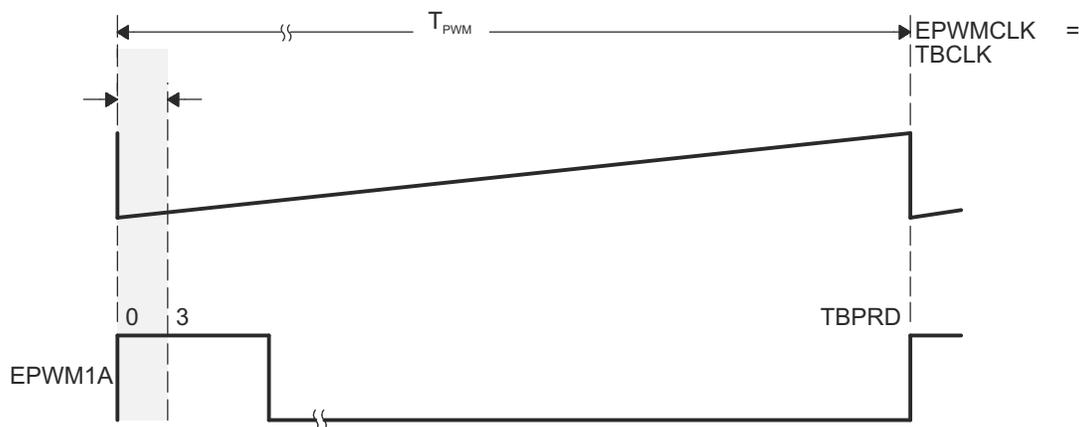
---

#### 13.15.1.5.3 Duty Cycle Range Limitation

In high-resolution mode, the MEP is not active for 100% of the PWM period and 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 using 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 or DBFED (the register corresponding to the edge with high-resolution displacement) must be greater than or equal to 7.

Duty cycle range limitations are illustrated in [Figure 13-85](#) to [Figure 13-88](#). 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 cannot 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 13-18](#). When high-resolution period control is enabled (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range; otherwise, there can be undefined behavior on the ePWMxA output.



**Figure 13-85. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

**Table 13-18. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles**

PWM Frequency <sup>(1)</sup> (kHz)	3 Cycles Minimum Duty	3 Cycles Maximum Duty <sup>(2)</sup>
200	0.6%	99.4%
400	1.2%	98.8%
600	1.8%	98.2%
800	2.4%	97.6%
1000	3%	97%
1200	3.6%	96.4%
1400	4.2%	95.8%
1600	4.8%	95.2%
1800	5.4%	94.6%
2000	6%	94%

(1) EPWMCLK = TBCLK = 100MHz

(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 13-86](#). In this configuration, the minimum duty cycle limitation is no longer an issue. However, there is a maximum duty limitation with same percent numbers as given in [Table 13-18](#).

#### CAUTION

If the application has enabled high-resolution period control (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range; otherwise, there can be undefined behavior on the ePWM output.

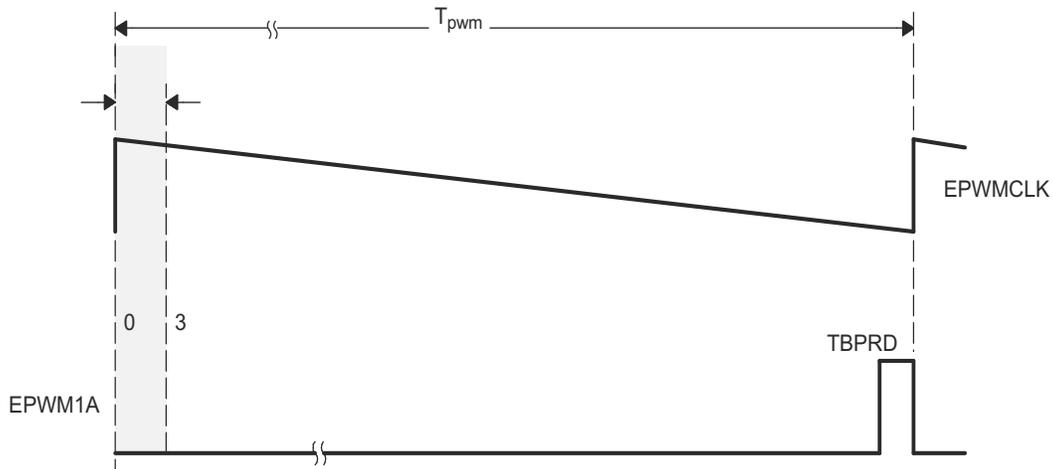


Figure 13-86. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)

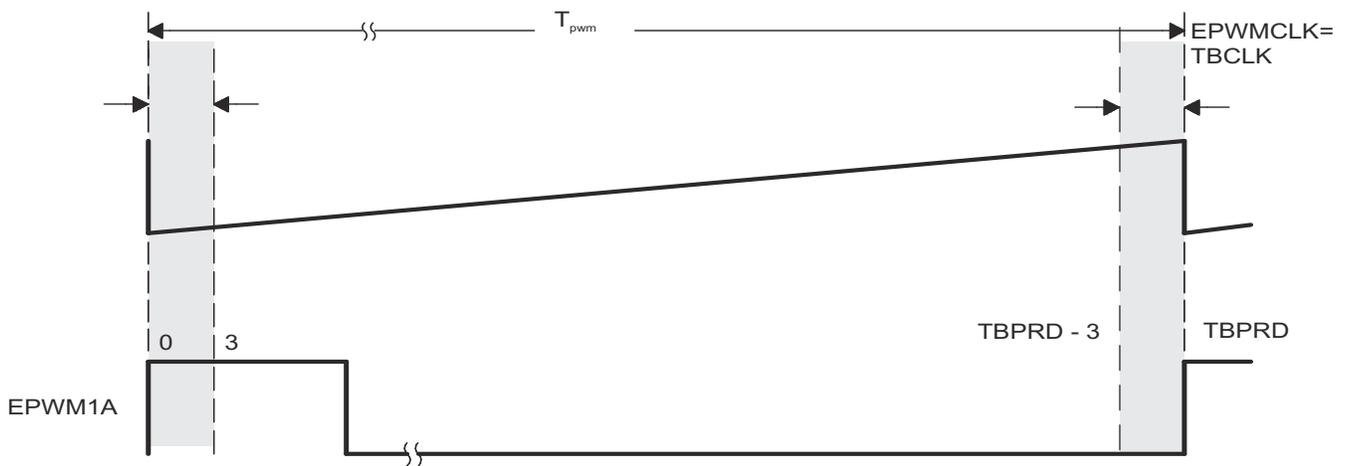


Figure 13-87. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1)

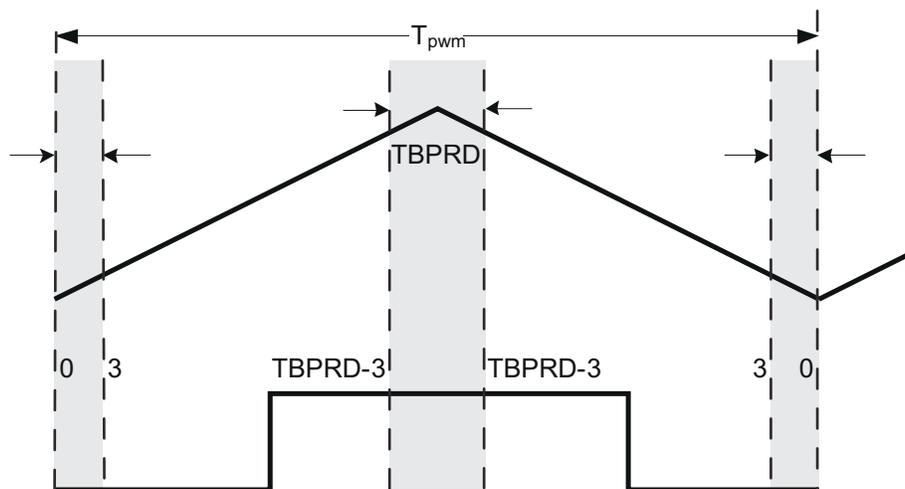


Figure 13-88. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 1)

#### 13.15.1.5.4 High-Resolution Period

High-resolution period control using the MEP logic is supported on devices with a Type 1 ePWM module or greater.

#### Note

When high-resolution period control is enabled, on ePWMxA only, and not ePWMxB output and conversely, the non high-resolution output has  $\pm 1$  TBCLK cycle jitter in up-count mode and  $\pm 2$  TBCLK cycle jitter in up-down count mode.

The scaling procedure described for duty cycle in [Section 13.15.1.5.2](#) applies for high-resolution period as well:

#### Assumptions for this example:

TBCLK	= 10ns (100MHz)
Required PWM frequency	= 175kHz (period of 571.428)
Number of MEP steps per coarse step at 180ps (10ns/180ps), (MEP_ScaleFactor)	= 55
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 = 174.82kHz (period =  $(571+1) * T_{TBCLK}$ ).
- If TBPRD = 570, then PWM frequency = 175.13kHz (period =  $(570+1) * T_{TBCLK}$ ).

In up-down count mode:

- If TBPRD = 286, then PWM frequency = 174.82kHz (period =  $(286*2) * T_{TBCLK}$ ).
- If TBPRD = 285, then PWM frequency = 175.44kHz (period =  $(285*2) * T_{TBCLK}$ ).

#### Solution:

With 55 MEP steps per coarse step at 180ps each:

#### Step 1: Percentage Integer Period value conversion for TBPRD register

$$\begin{aligned} \text{Integer period value} &= 571 * T_{TBCLK} \\ &= \text{int}(571.428) * T_{TBCLK} \\ &= \text{int}(\text{PWMperiod}) * T_{TBCLK} \end{aligned}$$

In up-count mode:

$$\begin{aligned} \text{TBPRD} &= 570 \text{ (TBPRD = period value - 1)} \\ &= 023Ah \end{aligned}$$

In up-down count mode:

$$\begin{aligned} \text{TBPRD} &= 285 \text{ (TBPRD = period value/2)} \\ &= 011Dh \end{aligned}$$

**Step 2: Fractional value conversion for TBPRDHR register**

In up-count mode:

$$\text{TBPRDHR register value} = (\text{frac}(\text{PWMperiod}) * \text{MEP\_ScaleFactor} + 0.5)$$

If auto-conversion enabled and HRMSTEP =

$$\text{MEP\_ScaleFactor value (55):} = \text{frac}(\text{PWMperiod}) \ll 8 \text{ (Shifting is to move the value to the high byte of TBPRDHR)}$$

$$\begin{aligned} \text{TBPRDHR register value} &= \text{frac}(571.428) \ll 8 \\ &= 0.428 \times 256 \\ &= 6D00\text{h} \end{aligned}$$

The auto-conversion then automatically performs the calculation, such that TBPRDHR MEP delay is scaled by hardware to:

$$\begin{aligned} &= ((\text{TBPRDHR}(15:0) \gg 8) \times \text{HRMSTEP} + 80\text{h}) \ll 8 \\ &= (006D\text{h} \times 55 + 80\text{h}) \gg 8 \\ &= (17EB\text{h}) \gg 8 \end{aligned}$$

$$\text{Period MEP delay} = 0017\text{h MEP Steps}$$

In up-down count mode:

$$\text{TBPRDHR register value} = (\text{frac}(\text{PWMperiod}) * \text{MEP\_ScaleFactor} + 0.5)$$

If auto-conversion enabled and HRMSTEP =

$$\text{MEP\_ScaleFactor value (55):} = \text{frac}(\text{PWMperiod} / 2) \ll 8 \text{ (Shifting is to move the value to the high byte of TBPRDHR)}$$

$$\begin{aligned} \text{TBPRDHR register value} &= \text{frac}(285.714) \ll 8 \\ &= 0.714 \times 256 \\ &= B600\text{h} \end{aligned}$$

The auto-conversion then automatically performs the calculation, such that TBPRDHR MEP delay is scaled by hardware to:

$$\begin{aligned} &= ((\text{TBPRDHR}(15:0) \gg 8) \times \text{HRMSTEP} + 80\text{h}) \ll 8 \\ &= (00B6\text{h} \times 55 + 80\text{h}) \gg 8 \\ &= (279A\text{h}) \gg 8 \end{aligned}$$

$$\text{Period MEP delay} = 0027\text{h MEP Steps}$$

#### 13.15.1.5.4.1 High-Resolution Period Configuration

To use high-resolution period, the ePWMx module must be initialized in the exact order presented.

The following steps use CMPA with shadow registers and the corresponding HRCNFG bits for high-resolution operation on EPWMxA. For high-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 can 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 using the SFO() function described in [Section 13.15.2](#).
11. To control high-resolution period, write to the TBPRDHR(M) registers.

---

#### Note

When high-resolution period mode is enabled, an EPWMxSYNC pulse introduces  $\pm 1$ -2 cycle jitter to the PWM ( $\pm 1$  cycle in up-count mode and  $\pm 2$  cycle in up-down count mode). For this reason, EPWMxSYNCO source cannot be set to CTR = 0 or CTR = CMPB. Otherwise, the jitter occurs on every PWM cycle with the synchronization pulse.

When EPWMxSYNCl is EPWMxSYNCO source, a software synchronization pulse can be issued only once during high-resolution period initialization. If a software sync pulse is applied while the PWM is running, the jitter appears on the PWM output at the time of the sync pulse.

---

### 13.15.1.6 Deadband High-Resolution Operation

---

#### Note

In up-count mode, the dead-band module is not available when any high-resolution mode is enabled.

---

#### Assumptions for this example:

System clock	= 10ns (100MHz)
Deadband enabled in half-cycle mode, TBCLK = EPWMCLK	
Required PWM frequency	1.33MHz (1/750ns)
Required PWM duty cycle	0.5 (50%)
Required Deadband Rising-Edge Delay	5% over duty
Required Deadband Rising-Edge Delay in ns	$(0.05 * 375ns) = 18.75ns$

---

#### Note

Similar to the duty cycle restrictions when using HRPWM, the DBRED and DBFED values must be greater than 3 to use high-resolution 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 (FED) and rising-edge delay (RED) becomes:

$$FED = DBFED * TBCLK / 2$$

$$RED = DBRED * TBCLK / 2$$

#### DBRED and DBFED calculated values:

Required Deadband Rising-Edge Delay in ns = 18.75ns

$$DBRED = RED / (TBCLK / 2)$$

$$DBRED = 18.75ns/5ns$$

$$DBRED \text{ Required} = 3.75ns$$

With 55 MEP steps per coarse step at 180ps each:

**Step 1: Integer Deadband value conversion for DBREDM register**

Integer DBRED value	= int (RED / (TBCLK / 2))
	= int (3.75)
DBRED	= 3

**Step 2: Fractional value conversion for Deadband 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 ignores 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 = frac((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.

---

**13.15.1.7 Scale Factor Optimizing Software (SFO)**

The micro edge positioner (MEP) logic is capable of placing an edge in one of 255 discrete time steps. As previously mentioned, the size of these steps is on the order of 150ps (see the device 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 can use the TI-supplied MEP scale factor optimization (SFO) software function. The SFO function helps to dynamically determine the number of MEP steps per EPWMCLK period while the HRPWM is in operation.

To utilize the MEP capabilities effectively, the correct value for the MEP scaling factor needs to be known by the software. To accomplish this, the HRPWM module has built in self-check and diagnostic capabilities that can be used to determine the optimum MEP scale factor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the optimum MEP scale factor. As such, MEP control and diagnostics registers are reserved for TI use.

A detailed description of the SFO library - SFO\_TI\_Build\_V8.lib software can be found in [SFO Library Software - SFO TI\\_Build\\_V8.lib](#).

### 13.15.1.8 HRPWM Examples Using Optimized Assembly Code

The best way to understand how to use the HRPWM capabilities is through two real examples:

1. Simple buck converter using asymmetrical PWM (count-up) with active high polarity.
2. DAC function using simple R+C reconstruction filter.

The following examples all have initialization and configuration code written in C. To make these easier to understand, the #defines shown below are used.

[Example 13-2](#) assumes MEP step size of 150ps and does not use the SFO library.

#### Example 13-2. #Defines for HRPWM Header Files

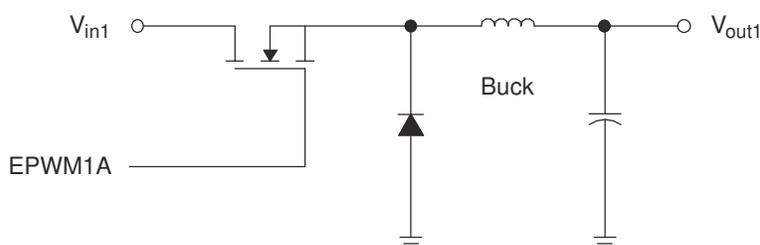
```
// HRPWM (High Resolution PWM) //
=====
// HRCNFG
#define HR_Disable 0x0
#define HR_REP 0x1 // Rising Edge position
#define HR_FEP 0x2 // Falling Edge position
#define HR_BEP 0x3 // Both Edge position #define HR_CMP 0x0 // CMPAHR controlled
#define HR_PHS 0x1 // TBPHSHR controlled #define HR_CTR_ZERO 0x0 // CTR = Zero event
#define HR_CTR_PRD 0x1 // CTR = Period event
#define HR_CTR_ZERO_PRD 0x2 // CTR = ZERO or Period event
#define HR_NORM_B 0x0 // Normal ePWMxB output
#define HR_INVERT_B 0x1 // ePWMxB is inverted ePWMxA output
```

### 13.15.1.8.1 Implementing a Simple Buck Converter

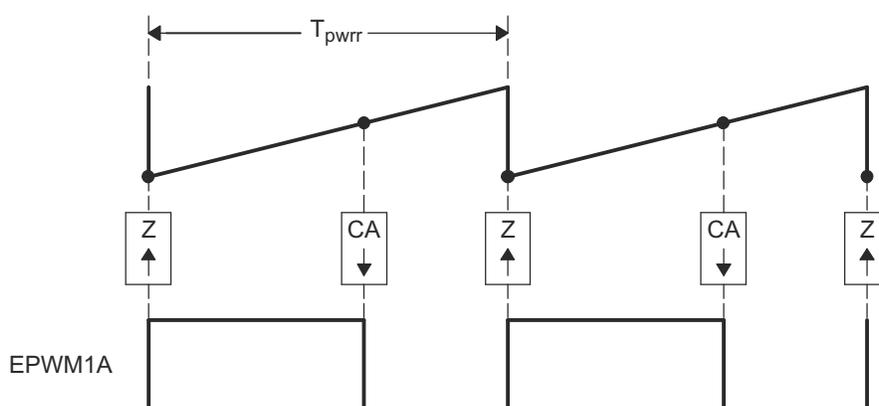
In this example, the PWM requirements are:

- PWM frequency = 1MHz (that is, TBPRD = 100)
- PWM mode = asymmetrical, up-count
- Resolution = 12.7 bits (with a MEP step size of 150ps)

Figure 13-89 and Figure 13-90 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 13-89. Simple Buck Controlled Converter Using a Single PWM**



**Figure 13-90. 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 13-3 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

This example assumes MEP step size of 150ps and does not use the SFO library.

Example 13-4 shows an assembly example of run-time code for the HRPWM buck converter.

### Example 13-3. HRPWM Buck Converter Initialization Code

```

void HrBuckDrvCnf(void)
{
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDL = TB_IMMEDIATE;           // set Immediate load
EPwm1Regs.TBPRD = 100;                             // Period set for 1000kHz 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.EPWSYNCOUTEN.all = 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 100MHZ
                                                // Note: Use SFO functions to update
                                                // MEP_ScaleFactor dynamically
}

```

### Example 13-4. HRPWM Buck Converter Run-Time Code

```

EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;=====
HRBUCK_DRV; (can execute within an ISR or loop)
;=====
    MOVW DP, #_HRBUCK_In
    MOVL XAR2,@_HRBUCK_In      ; Pointer to Input Q15 Duty (XAR2)
    MOVL XAR3,#CMPAHR1        ; Pointer to HRPWM CMPA reg (XAR3)

; Output for EPWM1A (HRPWM)
    MOV T,*XAR2 ; T <= Duty
    MPYU ACC,T,@_hrbuck_period ; Q15 to Q0 scaling based on Period
    MOV T,@_MEP_ScaleFactor    ; MEP scale factor (from optimizer s/w)
    MPYU P,T,@AL               ; P <= T * AL, Optimizer scaling
    MOVH @AL,P                 ; AL <= P, move result back to ACC
    ADD ACC, #0x080            ; MEP range and rounding adjustment
    MOVL *XAR3,ACC             ; CMPA: CMPAHR(31:8) <= ACC

; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
    MOV *+XAR3[2],AH           ; Store ACCH to regular CMPB

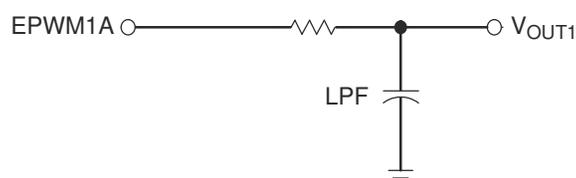
```

### 13.15.1.8.2 Implementing a DAC Function Using an R+C Reconstruction Filter

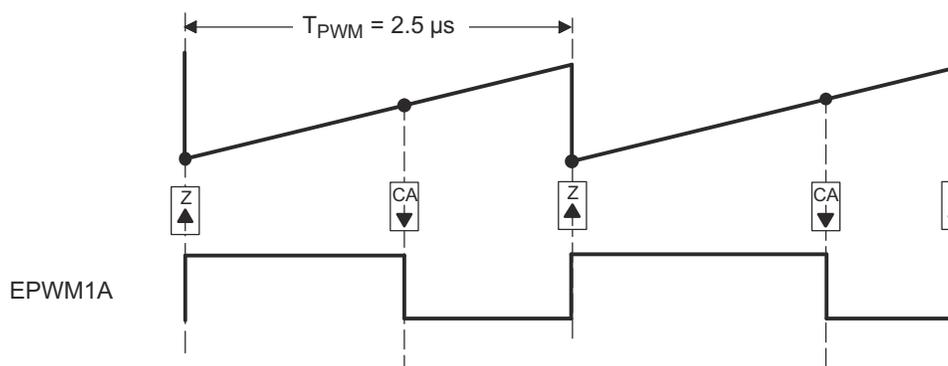
In this example, the PWM requirements are:

- PWM frequency = 400kHz (that is, TBPRD = 250)
- PWM mode = Asymmetrical, Up-count
- Resolution = 14 bits (MEP step size = 150ps)

Figure 13-91 and Figure 13-92 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 13-91. Simple Reconstruction Filter for a PWM-based DAC**



**Figure 13-92. 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 13-5](#) shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

[Example 13-6](#) shows an assembly example of run-time code that can execute in a high-speed ISR loop.

### Example 13-5. PWM DAC Function Initialization Code

```

void HrPwmDacDrvCnf(void)
{
// Config for conventional PWM first
EPwm1Regs.TBCTL.bit.PRDL = TB_IMMEDIATE;           // Set Immediate load
EPwm1Regs.TBPRD = 250;                             // Period set for 400kHz 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.EPWSYNCOUTEN.all = 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 100MHz.
                                                // Use SFO functions to update MEP_ScaleFactor
                                                // dynamically.
}

```

### Example 13-6. PWM DAC Function Run-Time Code

```

EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;=====
HRPWM_DAC_DRV; (can execute within an ISR or loop)
;=====
    MOVW DP, #_HRDAC_In
    MOVL XAR2,@_HRDAC_In           ; Pointer to input Q15 duty (XAR2)
    MOVL XAR3,#CMPAHR1            ; Pointer to HRPWM CMPA reg (XAR3)

; Output for EPWM1A (HRPWM)
    MOV T,*XAR2                   ; T <= duty
    MPY ACC,T,@_hrDAC_period       ; Q15 to Q0 scaling based on period
    ADD ACC,@_hrDAC_period<<15    ; Offset for bipolar operation
    MOV T,@_MEP_ScaleFactor        ; MEP scale factor (from optimizer s/w)
    MPYU P,T,@AL                  ; P <= T * AL, optimizer scaling
    MOVH @AL,P                    ; AL <= P, move result back to ACC
    ADD ACC,#0x080                ; MEP range and rounding adjustment
    MOVL *XAR3,ACC                ; CMPA: CMPAHR(31:8) <= ACC

; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
    MOV *+XAR3[2],AH              ; Store ACCH to regular CMPB

```

### 13.15.2 SFO Library Software - SFO\_TI\_Build\_V8.lib

Table 13-19 lists several features of the SFO\_TI\_Build\_V8.lib library.

**Table 13-19. SFO Library Features**

	SFO_TI_Build_V8.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

#### 13.15.2.1 Scale Factor Optimizer Function - int SFO()

This routine drives the micro-edge positioner (MEP) calibration module to run SFO diagnostics and determine the appropriate MEP scale factor (number of MEP steps per coarse EPWMCLK step) for a device at any given time.

If EPWMCLK = TBCLK = 100MHz and assuming the MEP step size is 150ps, the typical scale factor value at 100MHz = 66 MEP steps per TBCLK unit (10ns)

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 = 50MHz. MEP diagnostics logic uses EPWMCLK and not TBCLK, so the EPWMCLK restriction is an important constraint. Below 50MHz with device process variation, the MEP step size can decrease under cold temperature and high core voltage conditions to such a point that 255 MEP steps do 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 returns 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 maintains 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 13.15.1.5.3](#)).
- The SFO() function also updates the HRMSTEP register with the scale factor result. If the HRCNFG[AUTOCONV] bit is set, the application software is responsible only for setting  $CMPAHR = \text{fraction}(\text{PWMduty} * \text{PWMperiod}) \ll 8$  or  $CMPBHR = \text{fraction}(\text{PWMduty} * \text{PWMperiod}) \ll 8$  or  $TBPRDHR = \text{fraction}(\text{PWMperiod})$  while running SFO() in the background. The MEP Calibration Module then uses 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 needs to perform the necessary calculations manually so that:
  - $CMPAHR = (\text{fraction}(\text{PWMduty} * \text{PWMperiod}) * \text{MEP Scale Factor}) \ll 8 + 0x080$ .
  - Similar behavior applies for TBPHSHR, CMPBHR, DBREDHR, and DBFEDHR. Auto-conversion must be enabled when using TBPRDHR.

The following code snippet shows how to use the HRPWM DUTY using driverlib functions.

```
float32_t dutyFine = 85.62;
float32_t count = (dutyFine * (float32_t)(EPWM_TIMER_TBPRD << 8))/100;
uint32_t compCount = (count);
HRPWM_setCounterCompareValue(EPWM1_BASE, HRPWM_COUNTER_COMPARE_A, compCount);
HRPWM_setCounterCompareValue(EPWM1_BASE, HRPWM_COUNTER_COMPARE_B, compCount);
```

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 is often sufficient to execute the SFO function once every 5 to 10 seconds. If more rapid variations are expected, then execution can be performed more frequently to match the application. Note there is no high limit restriction on the SFO function repetition rate; hence, the SFO function can execute as quickly as the background loop is capable.

While using the HRPWM feature, HRPWM logic is not active for the first 3 EPWMCLK cycles of the PWM period (and the last 3 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 the 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 3 or above TBPRD-3. This can avoid any unexpected transitions on the PWM signal.

### 13.15.2.2 Software Usage

The software library function SFO(), calculates the MEP scale factor for the HRPWM-supported ePWM modules. The scale factor is an integer value in the range 1-255, and represents the number of micro step edge positions available for a system clock period. The scale factor value is returned in an integer variable called MEP\_ScaleFactor. For example, see [Table 13-20](#).

**Table 13-20. Factor Values**

Software Function call	Functional Description	Updated Variables
SFO()	Returns MEP scale factor in the HRMSTEP register	MEP_ScaleFactor and HRMSTEP register.

To use the HRPWM feature of the ePWMs, it is recommended that the SFO function be used as described here.

#### Step 1. Add "Include" Files

The SFO\_V8.h file needs to be included as follows. This include file is mandatory while using the SFO library function. For the SFO() to operate, the appropriate (Device)\_Device.h and (Device)\_Epwm\_defines.h must be included in the project. These include files are optional if customized header files are used in the end applications.

#### Example 13-7. A Sample of How to Add "Include" Files

```
#include "F28x7x_Device.h" // F28x7x Headerfile
#include "F28x7x_EPwm_defines.h" // init defines
#include "SFO_V8.h" // SFO lib functions (needed for HRPWM)
```

## Step 2. Element Declaration

Declare an integer variable for the scale factor value as shown below.

### Example 13-8. Declaring an Element

```
int MEP_ScaleFactor = 0; //scale factor value
volatile struct EPWM_REGS *ePWM[] = {0, &EPwm1Regs, &EPwm2Regs, &EPwm3Regs,
&EPwm4Regs};
```

## Step 3. MEP\_ScaleFactor Initialization

The SFO() function does not require a starting scale factor value in MEP\_ScaleFactor. Prior to using the MEP\_ScaleFactor variable in application code, SFO() can 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 13-9. Initializing With a Scale Factor Value

```
MEP_ScaleFactor initialized using function SFO ()
while (SFO() == 0) {} // MEP_ScaleFactor calculated by MEP Cal Module
```

## Step 4. Application Code

While the application is running, fluctuations in both device temperature and supply voltage can be expected. To be sure that good Scale Factors are used for each ePWM module, the SFO function can 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 13-10. SFO Function Calls

```
main ()
{
    int status;
    // User code
    // ePWM1, 2, 3, 4 are running in HRPWM mode
    // The status variable returns 1 once a new MEP_ScaleFactor has been
    // calculated by the MEP Calibration Module running SFO
    // diagnostics.
    status = SFO();
    if(status==2) {ESTOP0;} // The function returns a 2 if MEP_ScaleFactor is greater
    // than the maximum 255 allowed (error condition)
}
```

## 13.16 Software

### 13.16.1 EPWM Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/epwm

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 13.16.1.1 ePWM Trip Zone

FILE: epwm\_ex1\_trip\_zone.c

This example configures ePWM1 and ePWM2 as follows

- ePWM1 has TZ1 as one shot trip source
- ePWM2 has TZ1 as cycle by cycle trip source

Initially tie TZ1 high. During the test, monitor ePWM1 or ePWM2 outputs on a scope. Pull TZ1 low to see the effect.

##### *External Connections*

- ePWM1A is on GPIO0
- ePWM2A is on GPIO2
- TZ1 is on GPIO12

This example also makes use of the Input X-BAR. GPIO12 (the external trigger) is routed to the input X-BAR, from which it is routed to TZ1.

The TZ-Event is defined such that ePWM1A will undergo a One-Shot Trip and ePWM2A will undergo a Cycle-By-Cycle Trip.

#### 13.16.1.2 ePWM Up Down Count Action Qualifier

FILE: epwm\_ex2\_updown\_aq.c

This example configures ePWM1, ePWM2, ePWM3 to produce a waveform with independent modulation on ePWMxA and ePWMxB.

The compare values CMPA and CMPB are modified within the ePWM's ISR.

The TB counter is in up/down count mode for this example.

View the ePWM1A/B(GPIO0 & GPIO1), ePWM2A/B(GPIO2 & GPIO3) and ePWM3A/B(GPIO4 & GPIO5) waveforms on oscilloscope.

#### 13.16.1.3 ePWM Synchronization

FILE: epwm\_ex3\_synchronization.c

This example configures ePWM1, ePWM2, ePWM3 and ePWM4 as follows

- ePWM1 without phase shift as sync source
- ePWM2 with phase shift of 300 TBCLKs
- ePWM3 with phase shift of 600 TBCLKs
- ePWM4 with phase shift of 900 TBCLKs

##### *External Connections*

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO2 EPWM2A
- GPIO3 EPWM2B
- GPIO4 EPWM3A
- GPIO5 EPWM3B

- GPIO6 EPWM4A
- GPIO7 EPWM4B

#### Watch Variables

- None.

#### 13.16.1.4 ePWM Digital Compare

FILE: epwm\_ex4\_digital\_compare.c

This example configures ePWM1 as follows

- ePWM1 with DCAEVT1 forcing the ePWM output LOW
- GPIO24 is used as the input to the INPUT XBAR INPUT1
- INPUT1 (from INPUT XBAR) is used as the source for DCAEVT1
- GPIO24's PULL-UP resistor is enabled, in order to test the trip, PULL this pin to GND

#### External Connections

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO24 TZ1, pull this pin low to trip the ePWM

#### Watch Variables

- None.

#### 13.16.1.5 ePWM Digital Compare Event Filter Blanking Window

FILE: epwm\_ex5\_digital\_compare\_event\_filter.c

This example configures ePWM1 as follows

- ePWM1 with DCAEVT1 forcing the ePWM output LOW
- GPIO24 is used as the input to the INPUT XBAR INPUT1
- INPUT1 (from INPUT XBAR) is used as the source for DCAEVT1
- GPIO24's PULL-UP resistor is enabled, in order to test the trip, PULL this pin to GND
- ePWM1 with DCBEVT1 forcing the ePWM output LOW
- GPIO24 is used as the input to the INPUT XBAR INPUT1
- INPUT1 (from INPUT XBAR) is used as the source for DCAEVT1
- GPIO24's PULL-UP resistor is enabled, in order to test the trip, PULL this pin to GND
- DCBEVT1 uses the filtered version of DCBEVT1
- The DCFILT signal uses the blanking window to ignore the DCBEVT1 for the duration of DC Blanking window

#### External Connections

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO24 TRIPIN1, pull this pin low to trip the ePWM

#### Watch Variables

- None.

#### 13.16.1.6 ePWM Valley Switching

FILE: epwm\_ex6\_valley\_switching.c

This example configures ePWM1 as follows

- ePWM1 with DCAEVT1 forcing the ePWM output LOW
- GPIO24 is used as the input to the INPUT XBAR INPUT1
- INPUT1 (from INPUT XBAR) is used as the source for DCAEVT1
- GPIO24 is set to output and toggled in the main loop to trip the PWM
- ePWM1 with DCBEVT1 forcing the ePWM output LOW
- GPIO24 is used as the input to the INPUT XBAR INPUT1

- INPUT1 (from INPUT XBAR) is used as the source for DCAEVT1
- GPIO24 is set to output and toggled in the main loop to trip the PWM
- DCBEVT1 uses the filtered version of DCBEVT1
- The DCFILT signal uses the valley switching module to delay the
- DCFILT signal by a software defined DELAY value.

#### *External Connections*

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO24 TRIPIN1 (Output Pin, toggled through software)

#### *Watch Variables*

- None.

### **13.16.1.7 ePWM Digital Compare Edge Filter**

FILE: epwm\_ex7\_edge\_filter.c

This example configures ePWM1 as follows

- ePWM1 with DCBEVT2 forcing the ePWM output LOW as a CBC source
- GPIO24 is used as the input to the INPUT XBAR INPUT1
- INPUT1 (from INPUT XBAR) is used as the source for DCBEVT2
- GPIO24 is set to output and toggled in the main loop to trip the PWM
- The DCBEVT2 is the source for DCFILT
- The DCFILT will count edges of the DCBEVT2 and generate a signal to to trip the ePWM on the 4th edge of DCBEVT2

#### *External Connections*

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO24 TRIPIN1 (Output Pin, toggled through software)

#### *Watch Variables*

- None.

### **13.16.1.8 ePWM Deadband**

FILE: epwm\_ex8\_deadband.c

This example configures ePWM1 through ePWM6 as follows

- ePWM1 with Deadband disabled (Reference)
- ePWM2 with Deadband Active High
- ePWM3 with Deadband Active Low
- ePWM4 with Deadband Active High Complimentary
- ePWM5 with Deadband Active Low Complimentary
- ePWM6 with Deadband Output Swap (switch A and B outputs)

#### *External Connections*

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO2 EPWM2A
- GPIO3 EPWM2B
- GPIO4 EPWM3A
- GPIO5 EPWM3B
- GPIO6 EPWM4A
- GPIO7 EPWM4B
- GPIO8 EPWM5A
- GPIO9 EPWM5B

- GPIO10 EPWM6A
- GPIO11 EPWM6B

#### *Watch Variables*

- None.

### **13.16.1.9 ePWM Chopper**

FILE: epwm\_ex10\_chopper.c

This example configures ePWM1, ePWM2, ePWM3 and ePWM4 as follows

- ePWM1 with Chopper disabled (Reference)
- ePWM2 with chopper enabled at 1/8 duty cycle
- ePWM3 with chopper enabled at 6/8 duty cycle
- ePWM4 with chopper enabled at 1/2 duty cycle with One-Shot Pulse enabled

#### *External Connections*

- GPIO0 EPWM1A
- GPIO1 EPWM1B
- GPIO2 EPWM2A
- GPIO3 EPWM2B
- GPIO4 EPWM3A
- GPIO5 EPWM3B
- GPIO6 EPWM4A
- GPIO7 EPWM4B

#### *Watch Variables*

- None.

### **13.16.1.10 EPWM Configure Signal**

FILE: epwm\_ex11\_configure\_signal.c

This example configures ePWM1, ePWM2, ePWM3 to produce signal of desired frequency and duty. It also configures phase between the configured modules.

Signal of 10kHz with duty of 0.5 is configured on ePWMxA & ePWMxB with ePWMxB inverted. Also, phase of 120 degree is configured between ePWM1 to ePWM3 signals.

During the test, monitor ePWM1, ePWM2, and/or ePWM3 outputs on an oscilloscope.

- ePWM1A is on GPIO0
- ePWM1B is on GPIO1
- ePWM2A is on GPIO2
- ePWM2B is on GPIO3
- ePWM3A is on GPIO4
- ePWM3B is on GPIO5

### **13.16.1.11 Realization of Monoshot mode**

FILE: epwm\_ex12\_monoshot\_mode.c

This example showcases how to generate monoshot PWM output based on external trigger, that is, generating just a single pulse output on receipt of an external trigger. And the next pulse is generated only when the next trigger comes. The example utilizes external synchronization and T1 action qualifier event features to achieve the desired output.

ePWM1 is used to generate the monoshot output and ePWM2 is used as an external trigger for that. No external connections are required as ePWM2A is fed as the trigger using Input X-BAR automatically.

ePWM1 is configured to generate a single pulse of 0.5  $\mu$ s when received an external trigger. This is achieved by enabling the phase synchronization feature and configuring EPWMxSYNCl as EXTSYNCIN1. And this

EPWMxSYNCl is also configured as T1 event of action qualifier to set output HIGH while CTR = PRD action is used to set output LOW.

ePWM2 is configured to generate a 100 KHz signal with a duty of 1% (to simulate a rising edge trigger) which is routed to EXTSYNClN1 using Input XBAR.

Observe GPIO0 (EPWM1A : Monoshot Output) and GPIO2(EPWM2 : External Trigger) on oscilloscope.

*NOTE* : In the following example, the ePWM timer is still running in a continuous mode rather than a one-shot mode thus for more reliable implementation, refer to CLB based one shot PWM implementation demonstrated in "clb\_ex17\_one\_shot\_pwm" example

### 13.16.1.12 EPWM Action Qualifier (epwm\_up\_aq)

FILE: epwm\_ex13\_up\_aq.c

This example configures ePWM1, ePWM2, ePWM3 to produce an waveform with independent modulation on EPWMxA and EPWMxB.

The compare values CMPA and CMPB are modified within the ePWM's ISR.

The TB counter is in up count mode for this example.

View the EPWM1A/B(GPIO0 & GPIO1), EPWM2A/B(GPIO2 & GPIO3) and EPWM3A/B(GPIO4 & GPIO5) waveforms via an oscilloscope.

### 13.16.2 HRPWM Examples

*NOTE*: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/hrpwm

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 13.16.2.1 HRPWM Duty Control with SFO

FILE: hrpwm\_ex1\_duty\_sfo.c

This example modifies the MEP control registers to show edge displacement for high-resolution period with ePWM in Up count mode due to the HRPWM control extension of the respective ePWM module.

This example calls the following TI's MEP Scale Factor Optimizer (SFO) software library V8 functions:

*int SFO()*;

- updates MEP\_ScaleFactor dynamically when HRPWM is in use
- updates HRMSTEP register (exists only in EPwm1Regs register space) with MEP\_ScaleFactor value
- returns 2 if error: MEP\_ScaleFactor is greater than maximum value of 255 (Auto-conversion may not function properly under this condition)
- returns 1 when complete for the specified channel
- returns 0 if not complete for the specified channel

This example is intended to explain the HRPWM capabilities. The code can be optimized for code efficiency. Refer to TI's Digital power application examples and TI Digital Power Supply software libraries for details.

#### *External Connections*

- Monitor ePWM1 A/B pins on an oscilloscope.

#### 13.16.2.2 HRPWM Slider

FILE: hrpwm\_ex2\_slider.c

This example modifies the MEP control registers to show edge displacement due to HRPWM. Control blocks of the respective ePWM module channel A and B have fine edge movement due to HRPWM logic.

Monitor ePWM1 A/B pins on an oscilloscope.

### 13.16.2.3 HRPWM Period Control

FILE: hrpwm\_ex3\_prd\_updown\_sfo.c

This example modifies the MEP control registers to show edge displacement for high-resolution period with ePWM in Up-Down count mode due to the HRPWM control extension of the respective ePWM module.

This example calls the following TI's MEP Scale Factor Optimizer (SFO) software library V8 functions:

*int SFO();*

- updates MEP\_ScaleFactor dynamically when HRPWM is in use
- updates HRMSTEP register (exists only in EPwm1Regs register space) with MEP\_ScaleFactor value
- returns 2 if error: MEP\_ScaleFactor is greater than maximum value of 255 (Auto-conversion may not function properly under this condition)
- returns 1 when complete for the specified channel
- returns 0 if not complete for the specified channel

This example is intended to explain the HRPWM capabilities. The code can be optimized for code efficiency. Refer to TI's Digital power application examples and TI Digital Power Supply software libraries for details.

#### External Connections

- Monitor ePWM1 A/B pins on an oscilloscope.

### 13.16.2.4 HRPWM Duty Control with UPDOWN Mode

FILE: hrpwm\_ex4\_duty\_updown\_sfo.c

This example calls the following TI's MEP Scale Factor Optimizer (SFO) software library V8 functions:

*int SFO();*

- updates MEP\_ScaleFactor dynamically when HRPWM is in use
- updates HRMSTEP register (exists only in EPwm1Regs register space) with MEP\_ScaleFactor value
- returns 2 if error: MEP\_ScaleFactor is greater than maximum value of 255 (Auto-conversion may not function properly under this condition)
- returns 1 when complete for the specified channel
- returns 0 if not complete for the specified channel

This example is intended to explain the HRPWM capabilities. The code can be optimized for code efficiency. Refer to TI's Digital power application examples and TI Digital Power Supply software libraries for details.

#### External Connections

- Monitor ePWM1 A/B pins on an oscilloscope.

### 13.16.2.5 HRPWM Slider Test

FILE: hrpwm\_ex5\_slider\_qformat.c

This example modifies the MEP control registers to show edge displacement due to HRPWM. Control blocks of the respective ePWM module channel A and B will have fine edge movement due to HRPWM logic. Load the hrpwm\_slider.gel file. Select the HRPWM\_eval from the GEL menu. A FineDuty slider graphics will show up in CCS. Load the program and run. Use the Slider to and observe the EPWM edge displacement for each slider step change. This explains the MEP control on the EPwmxA channels.

Monitor ePWM1 A/B pins on an oscilloscope.

### 13.16.2.6 HRPWM Duty Up Count

FILE: hrpwm\_ex6\_duty\_sfo\_qformat.c

This example modifies the MEP control registers to show edge displacement for high-resolution period with ePWM in Up count mode due to the HRPWM control extension of the respective ePWM module.

This example calls the following TI's MEP Scale Factor Optimizer (SFO) software library V8 functions:

*int SFO();*

- updates MEP\_ScaleFactor dynamically when HRPWM is in use
- updates HRMSTEP register (exists only in EPwm1Regs register space) with MEP\_ScaleFactor value
- returns 2 if error: MEP\_ScaleFactor is greater than maximum value of 255 (Auto-conversion may not function properly under this condition)
- returns 1 when complete for the specified channel
- returns 0 if not complete for the specified channel

This example is intended to explain the HRPWM capabilities. The code can be optimized for code efficiency. Refer to TI's Digital power application examples and TI Digital Power Supply software libraries for details.

To run this example:

1. Run this example at maximum SYSCLKOUT
2. Activate Real time mode
3. Run the code

#### *External Connections*

- Monitor ePWM1 A/B pins on an oscilloscope.

#### *Watch Variables*

- status - Example run status
- updateFine - Set to 1 use HRPWM capabilities and observe in fine MEP steps(default) Set to 0 to disable HRPWM capabilities and observe in coarse SYSCLKOUT cycle steps

### **13.16.2.7 HRPWM Period Up-Down Count**

FILE: hrpwm\_ex7\_prd\_updown\_sfo\_qformat.c

This example modifies the MEP control registers to show edge displacement for high-resolution period with ePWM in Up-Down count mode due to the HRPWM control extension of the respective ePWM module.

This example calls the following TI's MEP Scale Factor Optimizer (SFO) software library V8 functions:

*int SFO();*

- updates MEP\_ScaleFactor dynamically when HRPWM is in use
- updates HRMSTEP register (exists only in EPwm1Regs register space) with MEP\_ScaleFactor value
- returns 2 if error: MEP\_ScaleFactor is greater than maximum value of 255 (Auto-conversion may not function properly under this condition)
- returns 1 when complete for the specified channel
- returns 0 if not complete for the specified channel

This example is intended to explain the HRPWM capabilities. The code can be optimized for code efficiency. Refer to TI's Digital power application examples and TI Digital Power Supply software libraries for details.

To run this example:

1. Run this example at maximum SYSCLKOUT
2. Activate Real time mode
3. Run the code

#### *External Connections*

- Monitor ePWM1 A/B pins on an oscilloscope.

#### *Watch Variables*

- updateFine - Set to 1 use HRPWM capabilities and observe in fine MEP steps(default) Set to 0 to disable HRPWM capabilities and observe in coarse SYSCLKOUT cycle steps

## 13.17 ePWM Registers

This section describes the Enhanced Pulse Width Modulator registers.

### 13.17.1 EPWM Base Address Table

**Table 13-21. EPWM Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
EPwm1Regs	EPWM_REGS	EPWM1_BASE	0x0000_4000	YES
EPwm2Regs	EPWM_REGS	EPWM2_BASE	0x0000_4100	YES
EPwm3Regs	EPWM_REGS	EPWM3_BASE	0x0000_4200	YES
EPwm4Regs	EPWM_REGS	EPWM4_BASE	0x0000_4300	YES
EPwm5Regs	EPWM_REGS	EPWM5_BASE	0x0000_4400	YES
EPwm6Regs	EPWM_REGS	EPWM6_BASE	0x0000_4500	YES
EPwm7Regs	EPWM_REGS	EPWM7_BASE	0x0000_4600	YES

### 13.17.2 EPWM\_REGS Registers

Table 13-22 lists the memory-mapped registers for the EPWM\_REGS registers. All register offset addresses not listed in Table 13-22 should be considered as reserved locations and the register contents should not be modified.

**Table 13-22. EPWM\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TBCTL	Time Base Control Register		<a href="#">Go</a>
1h	TBCTL2	Time Base Control Register 2		<a href="#">Go</a>
3h	EPWMSYNCINSEL	EPWMxSYNCIN Source Select Register		<a href="#">Go</a>
4h	TBCTR	Time Base Counter Register		<a href="#">Go</a>
5h	TBSTS	Time Base Status Register		<a href="#">Go</a>
6h	EPWMSYNCOUTEN	EPWMxSYNCOUT Source Enable Register		<a href="#">Go</a>
7h	TBCTL3	Time Base Control Register 3		<a href="#">Go</a>
8h	CMPCTL	Counter Compare Control Register		<a href="#">Go</a>
9h	CMPCTL2	Counter Compare Control Register 2		<a href="#">Go</a>
Ch	DBCTL	Dead-Band Generator Control Register		<a href="#">Go</a>
Dh	DBCTL2	Dead-Band Generator Control Register 2		<a href="#">Go</a>
10h	AQCTL	Action Qualifier Control Register		<a href="#">Go</a>
11h	AQTSRCSEL	Action Qualifier Trigger Event Source Select Register		<a href="#">Go</a>
14h	PCCTL	PWM Chopper Control Register		<a href="#">Go</a>
18h	VCAPCTL	Valley Capture Control Register		<a href="#">Go</a>
19h	VCNTCFG	Valley Counter Config Register		<a href="#">Go</a>
20h	HRCNFG	HRPWM Configuration Register	EALLOW	<a href="#">Go</a>
21h	HRPWR	HRPWM Power Register	EALLOW	<a href="#">Go</a>
26h	HRMSTEP	HRPWM MEP Step Register	EALLOW	<a href="#">Go</a>
27h	HRCNFG2	HRPWM Configuration 2 Register	EALLOW	<a href="#">Go</a>
2Dh	HRPCTL	High Resolution Period Control Register	EALLOW	<a href="#">Go</a>
2Eh	TRREM	HRPWM High Resolution Remainder Register	EALLOW	<a href="#">Go</a>
34h	GLDCTL	Global PWM Load Control Register	EALLOW	<a href="#">Go</a>
35h	GLDCFG	Global PWM Load Config Register	EALLOW	<a href="#">Go</a>
38h	EPWMXLINK	EPWMx Link Register		<a href="#">Go</a>
40h	AQCTLA	Action Qualifier Control Register For Output A		<a href="#">Go</a>
41h	AQCTLA2	Additional Action Qualifier Control Register For Output A		<a href="#">Go</a>
42h	AQCTLB	Action Qualifier Control Register For Output B		<a href="#">Go</a>
43h	AQCTLB2	Additional Action Qualifier Control Register For Output B		<a href="#">Go</a>
47h	AQSFRFC	Action Qualifier Software Force Register		<a href="#">Go</a>
49h	AQCSFRFC	Action Qualifier Continuous S/W Force Register		<a href="#">Go</a>
50h	DBREDHR	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		<a href="#">Go</a>
51h	DBRED	Dead-Band Generator Rising Edge Delay High Resolution Mirror Register		<a href="#">Go</a>
52h	DBFEDHR	Dead-Band Generator Falling Edge Delay High Resolution Register		<a href="#">Go</a>
53h	DBFED	Dead-Band Generator Falling Edge Delay Count Register		<a href="#">Go</a>
60h	TBPHS	Time Base Phase High		<a href="#">Go</a>

**Table 13-22. EPWM\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
62h	TBPRDHR	Time Base Period High Resolution Register		<a href="#">Go</a>
63h	TBPRD	Time Base Period Register		<a href="#">Go</a>
6Ah	CMPA	Counter Compare A Register		<a href="#">Go</a>
6Ch	CMPB	Compare B Register		<a href="#">Go</a>
6Fh	CMPC	Counter Compare C Register		<a href="#">Go</a>
71h	CMPD	Counter Compare D Register		<a href="#">Go</a>
74h	GLDCTL2	Global PWM Load Control Register 2		<a href="#">Go</a>
77h	SWVDELVAL	Software Valley Mode Delay Register		<a href="#">Go</a>
80h	TZSEL	Trip Zone Select Register	EALLOW	<a href="#">Go</a>
82h	TZDCSEL	Trip Zone Digital Comparator Select Register	EALLOW	<a href="#">Go</a>
84h	TZCTL	Trip Zone Control Register	EALLOW	<a href="#">Go</a>
85h	TZCTL2	Additional Trip Zone Control Register	EALLOW	<a href="#">Go</a>
86h	TZCTLDCA	Trip Zone Control Register Digital Compare A	EALLOW	<a href="#">Go</a>
87h	TZCTLDCB	Trip Zone Control Register Digital Compare B	EALLOW	<a href="#">Go</a>
8Dh	TZEINT	Trip Zone Enable Interrupt Register	EALLOW	<a href="#">Go</a>
93h	TZFLG	Trip Zone Flag Register		<a href="#">Go</a>
94h	TZCBCFLG	Trip Zone CBC Flag Register		<a href="#">Go</a>
95h	TZOSTFLG	Trip Zone OST Flag Register		<a href="#">Go</a>
97h	TZCLR	Trip Zone Clear Register	EALLOW	<a href="#">Go</a>
98h	TZCBCCLR	Trip Zone CBC Clear Register	EALLOW	<a href="#">Go</a>
99h	TZOSTCLR	Trip Zone OST Clear Register	EALLOW	<a href="#">Go</a>
9Bh	TZFRC	Trip Zone Force Register	EALLOW	<a href="#">Go</a>
A4h	ETSEL	Event Trigger Selection Register		<a href="#">Go</a>
A6h	ETPS	Event Trigger Pre-Scale Register		<a href="#">Go</a>
A8h	ETFLG	Event Trigger Flag Register		<a href="#">Go</a>
AAh	ETCLR	Event Trigger Clear Register		<a href="#">Go</a>
ACh	ETFRC	Event Trigger Force Register		<a href="#">Go</a>
A Eh	ETINTPS	Event-Trigger Interrupt Pre-Scale Register		<a href="#">Go</a>
B0h	ETSOCP	Event-Trigger SOC Pre-Scale Register		<a href="#">Go</a>
B2h	ETCNTINITCTL	Event-Trigger Counter Initialization Control Register		<a href="#">Go</a>
B4h	ETCNTINIT	Event-Trigger Counter Initialization Register		<a href="#">Go</a>
C0h	DCTRIPSEL	Digital Compare Trip Select Register	EALLOW	<a href="#">Go</a>
C3h	DCACTL	Digital Compare A Control Register	EALLOW	<a href="#">Go</a>
C4h	DCBCTL	Digital Compare B Control Register	EALLOW	<a href="#">Go</a>
C7h	DCFCTL	Digital Compare Filter Control Register	EALLOW	<a href="#">Go</a>
C8h	DCCAPCTL	Digital Compare Capture Control Register	EALLOW	<a href="#">Go</a>
C9h	DCFOFFSET	Digital Compare Filter Offset Register		<a href="#">Go</a>
CAh	DCFOFFSETCNT	Digital Compare Filter Offset Counter Register		<a href="#">Go</a>
CBh	DCFWINDOW	Digital Compare Filter Window Register		<a href="#">Go</a>
CCh	DCFWINDOWCNT	Digital Compare Filter Window Counter Register		<a href="#">Go</a>
CDh	BLANKPULSEMIXSEL	Blanking window trigger pulse select register	EALLOW	<a href="#">Go</a>
CFh	DCCAP	Digital Compare Counter Capture Register		<a href="#">Go</a>
D2h	DCAHTRIPSEL	Digital Compare AH Trip Select	EALLOW	<a href="#">Go</a>
D3h	DCALTRIPSEL	Digital Compare AL Trip Select	EALLOW	<a href="#">Go</a>

**Table 13-22. EPWM\_REGS Registers (continued)**

Offset	Acronym	Register Name	Write Protection	Section
D4h	DCBHTRIPSEL	Digital Compare BH Trip Select	EALLOW	<a href="#">Go</a>
D5h	DCBLTRIPSEL	Digital Compare BL Trip Select	EALLOW	<a href="#">Go</a>
FAh	EPWMLOCK	EPWM Lock Register		<a href="#">Go</a>
FDh	HWVDELVAL	Hardware Valley Mode Delay Register		<a href="#">Go</a>
FEh	VCNTVAL	Hardware Valley Counter Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 13-23](#) shows the codes that are used for access types in this section.

**Table 13-23. EPWM\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
WOnce	WOnce	Write Write once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 13.17.2.1 TBCTL Register (Offset = 0h) [Reset = 0083h]

TBCTL is shown in [Figure 13-93](#) and described in [Table 13-24](#).

Return to the [Summary Table](#).

Time Base Control Register

**Figure 13-93. 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	RESERVED		PRDL	PHSEN	CTRMODE	
R/W-1h	R-0/W1S-0h	R-0h		R/W-0h	R/W-0h	R/W-3h	

**Table 13-24. TBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events 00: Stop after the next time-base counter increment or decrement 01: Stop when counter completes a whole cycle: - Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD) - Down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00) - Up-down-count mode: stop when the time-base counter = 0x00 (TBCTR = 0x00) 1x: Free run Reset type: SYSRSn
13	PHSDIR	R/W	0h	Phase Direction Bit This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event. In the up-count and down-count modes this bit is ignored. 0: Count down after the synchronization event. 1: Count up after the synchronization event. Reset type: SYSRSn
12-10	CLKDIV	R/W	0h	Time Base Clock Pre-Scale Bits These bits select the time base clock pre-scale value (TBCLK = EPWMCLK/(HSPCLKDIV * CLKDIV): 000: /1 (default on reset) 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128 Reset type: SYSRSn

**Table 13-24. TBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-7	HSPCLKDIV	R/W	1h	High Speed Time Base Clock Pre-Scale Bits These bits determine part of the time-base clock prescale value. $TBCLK = EPWMCLK / (HSPCLKDIV \times CLKDIV)$ . This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral. 000: /1 001: /2 (default on reset) 010: /4 011: /6 100: /8 101: /10 110: /12 111: /14 Reset type: SYSRSn
6	SWFSYNC	R-0/W1S	0h	Software Forced Sync Pulse 0: Writing a 0 has no effect and reads always return a 0. 1: Writing a 1 forces a one-time synchronization pulse to be generated. SWFSYNC can be enabled to affect EPWMxSYNCO by setting the EPWMSYNCOOUTEN.SWEN bit. Reset type: SYSRSn
5-4	RESERVED	R	0h	Reserved
3	PRDL	R/W	0h	Active Period Reg Load from Shadow Select 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. A write/read to the TBPRD register accesses the shadow register. 1: Immediate Mode (Shadow register bypassed): A write or read to the TBPRD register accesses the active register. Reset type: SYSRSn
2	PHSEN	R/W	0h	Counter Reg Load from Phase Reg Enable 0: Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS). 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. Reset type: SYSRSn
1-0	CTRM	R/W	3h	Counter Mode 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: 00: Up-count mode 01: Down-count mode 10: Up-down count mode 11: Freeze counter operation (default on reset) Reset type: SYSRSn

### 13.17.2.2 TBCTL2 Register (Offset = 1h) [Reset = 0000h]

TBCTL2 is shown in [Figure 13-94](#) and described in [Table 13-25](#).

Return to the [Summary Table](#).

Time Base Control Register 2

**Figure 13-94. TBCTL2 Register**

15	14	13	12	11	10	9	8
PRDLDSYNC		RESERVED			RESERVED		
R/W-0h		R-0h			R-0-0h		
7	6	5	4	3	2	1	0
OSHTSYNC	OSHTSYNCMODE	RESERVED	RESERVED				
R-0/W1S-0h	R/W-0h	R/W-0h	R-0-0h				

**Table 13-25. TBCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	PRDLDSYNC	R/W	0h	Shadow to Active Period Register Load on SYNC event 00: Shadow to Active Load of TBPRD occurs only when TBCTR = 0 (same as legacy). 01: Shadow to Active Load of TBPRD occurs both when TBCTR = 0 and when SYNC occurs. 10: Shadow to Active Load of TBPRD occurs only when a SYNC is received. 11: Reserved Note: This bit selection is valid only if TBCTL[PRDLD]=0. Reset type: SYSRSn
13-12	RESERVED	R	0h	Reserved
11-8	RESERVED	R-0	0h	Reserved
7	OSHTSYNC	R-0/W1S	0h	Oneshot sync bit 0: Writing a '0' has no effect. 1: Allow one sync pulse to propagate. Reset type: SYSRSn
6	OSHTSYNCMODE	R/W	0h	Oneshot sync enable bit 0: Oneshot sync mode disabled 1: Oneshot sync mode enabled Reset type: SYSRSn
5	RESERVED	R/W	0h	Reserved
4-0	RESERVED	R-0	0h	Reserved

### 13.17.2.3 EPWMSYNCINSEL Register (Offset = 3h) [Reset = 0001h]

EPWMSYNCINSEL is shown in [Figure 13-95](#) and described in [Table 13-26](#).

Return to the [Summary Table](#).

EPWMxSYNCIN Source Select Register

**Figure 13-95. EPWMSYNCINSEL Register**

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

**Table 13-26. EPWMSYNCINSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4-0	SEL	R/W	1h	These bits determine the source of the EPWMxSYNCl signal. 0x00 Disabled Other Values defined in the 'ePWM SYNC Selection' table Reset type: SYSRSn

### 13.17.2.4 TBCTR Register (Offset = 4h) [Reset = 0000h]

TBCTR is shown in [Figure 13-96](#) and described in [Table 13-27](#).

Return to the [Summary Table](#).

Time Base Counter Register

**Figure 13-96. TBCTR Register**

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

**Table 13-27. TBCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBCTR	R/W	0h	Time Base Counter Register Reset type: SYSRSn

### 13.17.2.5 TBSTS Register (Offset = 5h) [Reset = 0001h]

TBSTS is shown in [Figure 13-97](#) and described in [Table 13-28](#).

Return to the [Summary Table](#).

Time Base Status Register

**Figure 13-97. TBSTS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					CTRMAX	SYNCI	CTDIR
R-0-0h					R/W1C-0h	R/W1C-0h	R-1h

**Table 13-28. TBSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-3	RESERVED	R-0	0h	Reserved
2	CTRMAX	R/W1C	0h	Time-Base Counter Max Latched Status Bit 0: Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect. 1: Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event. Reset type: SYSRSn
1	SYNCI	R/W1C	0h	Input Synchronization Latched Status Bit 0: Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred. 1: Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit will clear the latched event. Reset type: SYSRSn
0	CTDIR	R	1h	Time Base Counter Direction Status Bit 0: Time-Base Counter is currently counting down. 1: Time-Base Counter is currently counting up. Note: This bit is only valid when the counter is not frozen. Reset type: SYSRSn

### 13.17.2.6 EPWMSYNCOUEN Register (Offset = 6h) [Reset = 0001h]

EPWMSYNCOUEN is shown in [Figure 13-98](#) and described in [Table 13-29](#).

Return to the [Summary Table](#).

EPWMxSYNCOU Source Enable Register

**Figure 13-98. EPWMSYNCOUEN Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT1EN	DCAEVT1EN	CMPDEN	CMPDEN	CMPBEN	ZEROEN	SWEN
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h

**Table 13-29. EPWMSYNCOUEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	RESERVED	R	0h	Reserved
6	DCBEVT1EN	R/W	0h	This bit enables the DCBEVT1.sync event to set the EPWMxSYNCO signal. 0 Disabled 1 The EPWMxSYNCO signal is pulsed for one PWM clock period upon a DCBEVT1.sync event Reset type: SYSRSn
5	DCAEVT1EN	R/W	0h	This bit enables the DCAEVT1.sync event to set the EPWMxSYNCOU signal. 0 Disabled 1 The EPWMxSYNCOU signal is pulsed for one PWM clock period upon a DCAEVT1.sync event Reset type: SYSRSn
4	CMPDEN	R/W	0h	This bit enables the TBCTR = CMPD event to set the EPWMxSYNCO signal. 0 Disabled 1 The EPWMxSYNCO signal is pulsed for one PWM clock period upon a time-base counter equal to counter compare D event (TBCTR = CMPD) Reset type: SYSRSn
3	CMPDEN	R/W	0h	This bit enables the TBCTR = CMPC event to set the EPWMxSYNCO signal. 0 Disabled 1 The EPWMxSYNCO signal is pulsed for one PWM clock period upon a time-base counter equal to counter compare C event (TBCTR = CMPC) Reset type: SYSRSn
2	CMPBEN	R/W	0h	This bit enables the TBCTR = CMPB event to set the EPWMxSYNCO signal. 0 Disabled 1 The EPWMxSYNCO signal is pulsed for one PWM clock period upon a time-base counter equal to counter compare B event (TBCTR = CMPB) Reset type: SYSRSn
1	ZEROEN	R/W	0h	This bit enables the TBCTR = 0x0000 event to set the EPWMxSYNCOU signal. 0 Disabled 1 The EPWMxSYNCOU signal is pulsed for one PWM clock period upon the value of TBCTR changing to 0x0000 Reset type: SYSRSn

**Table 13-29. EPWMSYNCOUEN Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	SWEN	R/W	1h	This bit enables the TBCTL.SWFSYNC bit to set the EPWMxSYNCO signal. 0 Disabled 1 The EPWMxSYNCO signal is pulsed for one PWM clock period when the TBCTL.SWFSYNC bit is set Reset type: SYSRSn

### 13.17.2.7 TBCTL3 Register (Offset = 7h) [Reset = 0000h]

TBCTL3 is shown in [Figure 13-99](#) and described in [Table 13-30](#).

Return to the [Summary Table](#).

Time Base Control Register 3

**Figure 13-99. TBCTL3 Register**

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

**Table 13-30. TBCTL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	Reserved
0	OSSFRGEN	R/W	0h	This bit determines which bit sets the EPWMxSYNCOUT One Shot Latch. 0 TBCTL2[OSHTSYNC] sets the One Shot Latch 1 GLDCTL2[OSHTLD] sets the One Shot Latch Reset type: SYSRSn

### 13.17.2.8 CMPCTL Register (Offset = 8h) [Reset = 0000h]

CMPCTL is shown in [Figure 13-100](#) and described in [Table 13-31](#).

Return to the [Summary Table](#).

Counter Compare Control Register

**Figure 13-100. CMPCTL Register**

15	14	13	12	11	10	9	8
RESERVED		LOADBSYNC		LOADASYNC		SHDWBFULL	SHDWAFULL
R-0-0h		R/W-0h		R/W-0h		R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	SHDWBMODE	RESERVED	SHDWAMODE	LOADBMODE		LOADAMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 13-31. CMPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13-12	LOADBSYNC	R/W	0h	Shadow to Active CMPB Register Load on SYNC event 00: Shadow to Active Load of CMPB:CMPBHR occurs according to LOADBMODE (bits 1,0) (same as legacy) 01: Shadow to Active Load of CMPB:CMPBHR occurs both according to LOADBMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPB:CMPBHR occurs only when a SYNC is received 11: Reserved Note: This bit is valid only if CMPCTL[SHDWBMODE] = 0. Reset type: SYSRSn
11-10	LOADASYNC	R/W	0h	Shadow to Active CMPA Register Load on SYNC event 00: Shadow to Active Load of CMPA:CMPAHR occurs according to LOADAMODE (bits 1,0) (same as legacy) 01: Shadow to Active Load of CMPA:CMPAHR occurs both according to LOADAMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPA:CMPAHR occurs only when a SYNC is received 11: Reserved Note: This bit is valid only if CMPCTL[SHDWAMODE] = 0. Reset type: SYSRSn
9	SHDWBFULL	R	0h	Counter-compare B (CMPB) Shadow Register Full Status Flag This bit self clears once a loadstrobe occurs. 0: CMPB shadow register not full yet 1: Indicates the CMPB shadow register is full a CPU write will overwrite current shadow value Reset type: SYSRSn
8	SHDWAFULL	R	0h	Counter-compare A (CMPA) Shadow Register Full Status Flag The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag. This bit self clears once a load-strobe occurs. 0: CMPA shadow register not full yet 1: Indicates the CMPA shadow register is full, a CPU write will overwrite the current shadow value Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved

**Table 13-31. CMPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	SHDWBMODE	R/W	0h	Counter-compare B (CMPB) Register Operating Mode 0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register 1: Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved
4	SHDWAMODE	R/W	0h	Counter-compare A (CMPA) Register Operating Mode 0: Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register 1: Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action Reset type: SYSRSn
3-2	LOADBMODE	R/W	0h	Active Counter-Compare B (CMPB) Load From Shadow Select Mode This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1). 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Reset type: SYSRSn
1-0	LOADAMODE	R/W	0h	Active Counter-Compare A (CMPA) Load From Shadow Select Mode This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1). 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Reset type: SYSRSn

### 13.17.2.9 CMPCTL2 Register (Offset = 9h) [Reset = 0000h]

CMPCTL2 is shown in [Figure 13-101](#) and described in [Table 13-32](#).

Return to the [Summary Table](#).

Counter Compare Control Register 2

**Figure 13-101. CMPCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED		LOADDSYNC		LOADCSYNC		RESERVED	
R-0-0h		R/W-0h		R/W-0h		R-0-0h	
7	6	5	4	3	2	1	0
RESERVED	SHDWDMODE	RESERVED	SHDWCMODE	LOADDMODE		LOADCMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 13-32. CMPCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R-0	0h	Reserved
13-12	LOADDSYNC	R/W	0h	Shadow to Active CMPD Register Load on SYNC event 00: Shadow to Active Load of CMPD occurs according to LOADDMODE 01: Shadow to Active Load of CMPD occurs both according to LOADDMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPD occurs only when a SYNC is received 11: Reserved Note: This bit is valid only if CMPCTL2[SHDWDMODE] = 0. Reset type: SYSRSn
11-10	LOADCSYNC	R/W	0h	Shadow to Active CMPC Register Load on SYNC event 00: Shadow to Active Load of CMPC occurs according to LOADCMODE 01: Shadow to Active Load of CMPC occurs both according to LOADCMODE bits and when SYNC occurs 10: Shadow to Active Load of CMPC occurs only when a SYNC is received 11: Reserved Note: This bit is valid only if CMPCTL2[SHDWCMODE] = 0. Reset type: SYSRSn
9-7	RESERVED	R-0	0h	Reserved
6	SHDWDMODE	R/W	0h	Counter-Compare D Register Operating Mode 0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 1: Immediate mode - only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate Compare action. Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved
4	SHDWCMODE	R/W	0h	Counter-Compare C Register Operating Mode 0: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 1: Immediate mode - only the Active compare register is used. All writes/reads via the CPU directly access the Active register for immediate Compare action. Reset type: SYSRSn

**Table 13-32. 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. Reset type: SYSRSn
1-0	LOADCMODE	R/W	0h	Active Counter-Compare C (CMPC) Load from Shadow Select Mode 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Note: Has no effect in Immediate mode. Reset type: SYSRSn

### 13.17.2.10 DBCTL Register (Offset = Ch) [Reset = 0000h]

DBCTL is shown in [Figure 13-102](#) and described in [Table 13-33](#).

Return to the [Summary Table](#).

Dead-Band Generator Control Register

**Figure 13-102. 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	

**Table 13-33. DBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	HALFCYCLE	R/W	0h	Half Cycle Clocking Enable Bit 0: Full cycle clocking enabled. The dead-band counters are clocked at the TBCLK rate. 1: Half cycle clocking enabled. The dead-band counters are clocked at TBCLK*2. Reset type: SYSRSn
14	DEDB_MODE	R/W	0h	Dead Band Dual-Edge B Mode Control (S8 switch) 0: Rising edge delay applied to InA/InB as selected by S4 switch (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. Reset type: SYSRSn
13-12	OUTSWAP	R/W	0h	Dead Band Output Swap Control Bit 13 controls the S6 switch and bit 12 controls the S7 switch. 00: OutA and OutB signals are as defined by OUT-MODE bits. 01: OutA = A-path as defined by OUT-MODE bits. OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path). 10: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path). OutB = B-path as defined by OUT-MODE bits. 11: OutA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B signal path). OutB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A signal path). Reset type: SYSRSn
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). Reset type: SYSRSn

**Table 13-33. DBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	SHDWDBREDMODE	R/W	0h	<p>RED Dead-Band Load Mode</p> <p>0: Immediate mode. Only the active DBRED register is used. All writes/reads via the CPU directly access the active register for immediate 'RED dead-band action.'</p> <p>1: Shadow mode. Operates as a double buffer. All writes via the CPU access Shadow register. Default at Reset is Immediate mode (for compatibility with legacy).</p> <p>Reset type: SYSRSn</p>
9-8	LOADFEDMODE	R/W	0h	<p>Active DBFED Load from Shadow Select Mode</p> <p>00: Load on Counter = 0 (CNT_eq)</p> <p>01: Load on Counter = Period (PRD_eq)</p> <p>10: Load on either Counter = 0, or Counter = Period</p> <p>11: Freeze (no loads possible)</p> <p>Note: has no effect in Immediate mode.</p> <p>Reset type: SYSRSn</p>
7-6	LOADREDMODE	R/W	0h	<p>Active DBRED Load from Shadow Select Mode</p> <p>00: Load on Counter = 0 (CNT_eq)</p> <p>01: Load on Counter = Period (PRD_eq)</p> <p>10: Load on either Counter = 0, or Counter = Period</p> <p>11: Freeze (no loads possible)</p> <p>Note: has no effect in Immediate mode.</p> <p>Reset type: SYSRSn</p>
5-4	IN_MODE	R/W	0h	<p>Dead-Band Input Mode Control</p> <p>Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown. This allows you to select the input source to the falling-edge and rising-edge delay. To produce classical dead-band waveforms the default is EPWMxA In is the source for both falling and rising-edge delays.</p> <p>00: EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.</p> <p>01: EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>10: EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>11: EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.</p> <p>Reset type: SYSRSn</p>
3-2	POLSEL	R/W	0h	<p>Polarity Select Control</p> <p>Bit 3 controls the S3 switch and bit 2 controls the S2 switch. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter. These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0x0. Other enhanced modes are also possible, but not regarded as typical usage modes.</p> <p>00: Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).</p> <p>01: Active low complementary (ALC) mode. EPWMxA is inverted.</p> <p>10: Active high complementary (AHC). EPWMxB is inverted.</p> <p>11: Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.</p> <p>Reset type: SYSRSn</p>

**Table 13-33. DBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	OUT_MODE	R/W	0h	Dead-Band Output Mode Control Bit 1 controls the S1 switch and bit 0 controls the S0 switch. 00: DBM is fully disabled or by-passed. In this mode the POLSEL and IN-MODE bits have no effect. 01: Apath = InA (delay is by-passed for A signal path) Bpath = FED (Falling Edge Delay in B signal path) 10: Apath = RED (Rising Edge Delay in A signal path) Bpath = InB (delay is by-passed for B signal path) 11: DBM is fully enabled (i.e. both RED and FED active) Reset type: SYSRSn

### 13.17.2.11 DBCTL2 Register (Offset = Dh) [Reset = 0000h]

DBCTL2 is shown in [Figure 13-103](#) and described in [Table 13-34](#).

Return to the [Summary Table](#).

Dead-Band Generator Control Register 2

**Figure 13-103. DBCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED					SHDWDBCTLM ODE	LOADDBCTLMODE	
R-0-0h					R/W-0h	R/W-0h	

**Table 13-34. 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 to bits [5:0] of the DBCTL register are shadowed. All other bits still access the active register. Reset type: SYSRSn
1-0	LOADDBCTLMODE	R/W	0h	Active DBCTL Load from Shadow Select Mode 00: Load on Counter = 0 (CNT_eq) 01: Load on Counter = Period (PRD_eq) 10: Load on either Counter = 0, or Counter = Period 11: Freeze (no loads possible) Note: has no effect in Immediate mode Reset type: SYSRSn

### 13.17.2.12 AQCTL Register (Offset = 10h) [Reset = 0000h]

AQCTL is shown in [Figure 13-104](#) and described in [Table 13-35](#).

Return to the [Summary Table](#).

Action Qualifier Control Register

**Figure 13-104. 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	SHDWAQBMO DE	RESERVED	SHDWAQAMO DE	LDAQBMODE		LDAQAMODE	
R-0-0h	R/W-0h	R-0-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 13-35. 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 AQCTL[SHDWAQBMODE] = 1. Reset type: SYSRSn
9-8	LDAQASYNC	R/W	0h	Shadow to Active AQCTLA Register Load on SYNC event 00: Shadow to Active Load of AQCTLA occurs according to LDAQAMODE 01: Shadow to Active Load of AQCTLA occurs both according to LDAQAMODE bits and when SYNC occurs. 10: Shadow to Active Load of AQCTLA occurs only when a SYNC is received. 11: Reserved Note: This bit is valid only if AQCTL[SHDWAQAMODE] = 1. Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	SHDWAQBMODE	R/W	0h	Action Qualifier B Register operating mode 1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register. Reset type: SYSRSn
5	RESERVED	R-0	0h	Reserved
4	SHDWAQAMODE	R/W	0h	Action Qualifier A Register operating mode 1: Shadow mode - operates as a double buffer. All writes via the CPU access Shadow register. 0: Immediate mode - only the Active action qualifier register is used. All writes/reads via the CPU directly access the Active register. Reset type: SYSRSn

**Table 13-35. AQCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	LDAQBMODE	R/W	0h	Active Action Qualifier B Load from Shadow Select Mode 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Note: has no effect in Immediate mode. Reset type: SYSRSn
1-0	LDAQAMODE	R/W	0h	Active Action Qualifier A Load from Shadow Select Mode 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Freeze (no loads possible) Note: has no effect in Immediate mode. Reset type: SYSRSn

**13.17.2.13 AQTSRCSEL Register (Offset = 11h) [Reset = 0000h]**

 AQTSRCSEL is shown in [Figure 13-105](#) and described in [Table 13-36](#).

 Return to the [Summary Table](#).

Action Qualifier Trigger Event Source Select Register

**Figure 13-105. AQTSRCSEL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2SEL				T1SEL			
R/W-0h				R/W-0h			

**Table 13-36. AQTSRCSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-4	T2SEL	R/W	0h	T2 Event Source Select Bits 0000: DCAEVT1 0001: DCAEVT2 0010: DCBEVT1 0011: DCBEVT2 0100: TZ1 0101: TZ2 0110: TZ3 0111: EPWMxSYNCl 1000: DCEVTFILT Others: Reserved Reset type: SYSRSn
3-0	T1SEL	R/W	0h	T1 Event Source Select Bits 0000: DCAEVT1 0001: DCAEVT2 0010: DCBEVT1 0011: DCBEVT2 0100: TZ1 0101: TZ2 0110: TZ3 0111: EPWMxSYNCl 1000: DCEVTFILT Others: Reserved Reset type: SYSRSn

### 13.17.2.14 PCCTL Register (Offset = 14h) [Reset = 0000h]

PCCTL is shown in [Figure 13-106](#) and described in [Table 13-37](#).

Return to the [Summary Table](#).

PWM Chopper Control Register

**Figure 13-106. PCCTL Register**

15	14	13	12	11	10	9	8
RESERVED						CHPDUTY	
R-0-0h						R/W-0h	
7	6	5	4	3	2	1	0
CHPFREQ			OSHTWTH			CHPEN	
R/W-0h			R/W-0h			R/W-0h	

**Table 13-37. PCCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10-8	CHPDUTY	R/W	0h	Chopping Clock Duty Cycle 000: Duty = 1/8 (12.5%) 001: Duty = 2/8 (25.0%) 010: Duty = 3/8 (37.5%) 011: Duty = 4/8 (50.0%) 100: Duty = 5/8 (62.5%) 101: Duty = 6/8 (75.0%) 110: Duty = 7/8 (87.5%) 111: Reserved Reset type: SYSRSn
7-5	CHPFREQ	R/W	0h	Chopping Clock Frequency 000: Divide by 1 (no prescale, = 12.5 MHz at 100 MHz TBCLK) 001: Divide by 2 (6.25 MHz at 100 MHz TBCLK) 010: Divide by 3 (4.16 MHz at 100 MHz TBCLK) 011: Divide by 4 (3.12 MHz at 100 MHz TBCLK) 100: Divide by 5 (2.50 MHz at 100 MHz TBCLK) 101: Divide by 6 (2.08 MHz at 100 MHz TBCLK) 110: Divide by 7 (1.78 MHz at 100 MHz TBCLK) 111: Divide by 8 (1.56 MHz at 100 MHz TBCLK) Reset type: SYSRSn
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) Reset type: SYSRSn

**Table 13-37. PCCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CHPEN	R/W	0h	PWM-Chopping Enable 0: Disable (bypass) PWM chopping function 1: Enable chopping function Reset type: SYSRSn

### 13.17.2.15 VCAPCTL Register (Offset = 18h) [Reset = 0000h]

VCAPCTL is shown in [Figure 13-107](#) and described in [Table 13-38](#).

Return to the [Summary Table](#).

Valley Capture Control Register

**Figure 13-107. VCAPCTL Register**

15	14	13	12	11	10	9	8
RESERVED					EDGEFILTDLY SEL	VDELAYDIV	
R-0-0h					R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
VDELAYDIV	RESERVED		TRIGSEL			VCAPSTART	VCAPE
R/W-0h	R-0-0h		R/W-0h			R-0/W1S-0h	R/W-0h

**Table 13-38. VCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10	EDGEFILTDLYSEL	R/W	0h	Valley Switching Mode Delay Selection 0: No delay applied to the edge filter output 1: HWDELAYVAL delay applied to the edge filter output Reset type: SYSRSn
9-7	VDELAYDIV	R/W	0h	Valley Delay Mode Divide Enable 000: HWVDELVAL = SWVDELVAL 001: HWVDELVAL = VCNTVAL+SWVDELVAL 010: HWVDELVAL = VCNTVAL>>1+SWVDELVAL 011: HWVDELVAL = VCNTVAL>>2+SWVDELVAL 100: HWVDELVAL = VCNTVAL>>4+SWVDELVAL Note: Delay value between the consecutive edge captures can optionally be divided by using these bits. Reset type: SYSRSn
6-5	RESERVED	R-0	0h	Reserved
4-2	TRIGSEL	R/W	0h	Status of Numbered of Captured Events 000: Capture sequence is triggered by software via writes to VCAPCTL[VCAPSTART]. 001: Capture sequence is triggered by CNT_zero event. 010: Capture sequence is triggered by PRD_eq event. 011: Capture sequence is triggered by CNT_zero or PRD_eq event. 100: Capture sequence is triggered by DCAEVT1 event. 101: Capture sequence is triggered by DCAEVT2 event. 110: Capture sequence is triggered by DCBEVT1 event. 111: Capture sequence is triggered by DCBEVT2 event. Note: Valley capture sequence triggered by the selected event in this register field. Once the chosen event occurs the capture sequence is armed. Event captures occur based of the event chosen in DCFCTL[SRCSSEL] register. Note: Same event may not be chosen in both DCFCTL[SRCSSEL] and VCAPCTL[TRIGSEL] registers. Note: Once the chosen event in VCAPCTL[TRIGSEL] occurs, irrespective of the current capture status, capture sequence is retriggered. Reset type: SYSRSn

**Table 13-38. VCAPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	VCAPSTART	R-0/W1S	0h	Valley Capture Start 0: Writing a 0 has no effect 1: Trigger the capture sequence once if VCAPCTL[TRIGSEL]=0x0 Note: This bit is used to start valley capture sequence through software. VCAPCTL[TRIGSEL] has to be chosen for software trigger for this bit to have any effect. Writing of 1 will result in one capture sequence trigger. Reset type: SYSRSn
0	VCAPE	R/W	0h	Valley Capture Enable/Disable 0: Disabled 1: Enabled Reset type: SYSRSn

### 13.17.2.16 VCNTCFG Register (Offset = 19h) [Reset = 0000h]

VCNTCFG is shown in [Figure 13-108](#) and described in [Table 13-39](#).

Return to the [Summary Table](#).

Valley Counter Config Register

**Figure 13-108. VCNTCFG Register**

15	14	13	12	11	10	9	8
STOPEDGESTS	RESERVED			STOPEDGE			
R-0h	R-0-0h			R/W-0h			
7	6	5	4	3	2	1	0
STARTEDGESTS	RESERVED			STARTEDGE			
R-0h	R-0-0h			R/W-0h			

**Table 13-39. VCNTCFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	STOPEDGESTS	R	0h	Stop Edge Status Bit 0: Stop edge has not occurred 1: Stop edge occurred Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STOPEDGE occurs. Note: This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL] Reset type: SYSRSn
14-12	RESERVED	R-0	0h	Reserved
11-8	STOPEDGE	R/W	0h	Counter Stop Edge Selection Once the counter operation is armed, upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL] pulse - valley counter would stop counting upon the occurrence of chosen number of events thorough this bit field. Stop counting on occurrence of: 0000: Do not stop 0001 1st edge 0010: 2nd edge 0011: 3rd edge ... 1,1,1,1: 15th edge Reset type: SYSRSn
7	STARTEDGESTS	R	0h	Start Edge Status Bit 0: Start edge has not occurred 1: Start edge occurred Note: This bit is set only after the trigger sequence is armed (upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL]) and STARTEDGE occurs. Note: This bit is reset by the occurrence of the trigger pulse selected through VCAPCTL[TRIGSEL] Reset type: SYSRSn
6-4	RESERVED	R-0	0h	Reserved

**Table 13-39. VCNTCFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	STARTEDGE	R/W	0h	Counter Start Edge Selection Once the counter operation is armed, upon occurrence of trigger pulse selected through VCAPCTL[TRIGSEL] pulse - valley counter would start counting upon the occurrence of chosen number of events through this bit field. Start counting on occurrence of 0000: Do not start 0001: 1st edge 0010: 2nd edge 0011: 3rd edge ... 1111: 15th edge Reset type: SYSRSn

### 13.17.2.17 HRCNFG Register (Offset = 20h) [Reset = 0000h]

HRCNFG is shown in [Figure 13-109](#) and described in [Table 13-40](#).

Return to the [Summary Table](#).

HRPWM Configuration Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 13-109. HRCNFG Register**

15	14	13	12	11	10	9	8
RESERVED		RESERVED	HRLOADB		CTLMODEB	EDGMODEB	
R/W-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	

**Table 13-40. HRCNFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R/W	0h	Reserved
13	RESERVED	R-0	0h	Reserved
12-11	HRLOADB	R/W	0h	Shadow Mode Bit Selects the time event that loads the CMPBHR shadow value into the active register. 00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Reserved Reset type: SYSRSn
10	CTLMODEB	R/W	0h	Control Mode Bits Selects the register (CMP/TBPRD or TBPHS) that controls the MEP: 0: CMPBHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset) 1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode). Reset type: SYSRSn
9-8	EDGMODEB	R/W	0h	Edge Mode Bits Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic: 00: HRPWM capability is disabled (default on reset) 01: MEP control of rising edge (CMPBHR) 10: MEP control of falling edge (CMPBHR) 11: MEP control of both edges (TBPHSHR or TBPRDHR) Reset type: SYSRSn
7	SWAPAB	R/W	0h	Swap ePWM A & B Output Signals This bit enables the swapping of the A & B signal outputs. The selection is as follows: 0: ePWMxA and ePWMxB outputs are unchanged. 1: ePWMxA signal appears on ePWMxB output and ePWMxB signal appears on ePWMxA output. Reset type: SYSRSn

**Table 13-40. HRCNFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	AUTOCONV	R/W	0h	<p>Auto Convert Delay Line Value</p> <p>Selects whether the fractional duty cycle/period/phase in the CMPAHR/TBPRDHR/TBPHSHR register is automatically scaled by the MEP scale factor in the HRMSTEP register or manually scaled by calculations in application software. The SFO library function automatically updates the HRMSTEP register with the appropriate MEP scale factor.</p> <p>0: Automatic HRMSTEP scaling is disabled. 1: Automatic HRMSTEP scaling is enabled.</p> <p>If application software is manually scaling the fractional duty cycle, or phase (i.e. software sets <math>CMPAHR = (\text{fraction}(PWMduty * PWMperiod) * MEP \text{ Scale Factor}) \ll 8 + 0x080</math> for duty cycle), then this mode must be disabled.</p> <p>Reset type: SYSRSn</p>
5	SELOUTB	R/W	0h	<p>EPWMxB Output Select Bit</p> <p>This bit selects which signal is output on the ePWMxB channel output.</p> <p>The inversion will take the high resolution mode into account and the inverted signal will contain any high resolution modification. The inversion takes place as the last step in modifying the ePWMxB signal.</p> <p>0: ePWMxB output is normal. 1: ePWMxB output is inverted version of ePWMxA signal.</p> <p>Reset type: SYSRSn</p>
4-3	HRLOAD	R/W	0h	<p>Shadow Mode Bit</p> <p>Selects the time event that loads the CMPAHR shadow value into the active register.</p> <p>00: Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01: Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10: Load on either CTR = Zero or CTR = PRD 11: Reserved</p> <p>Reset type: SYSRSn</p>
2	CTLMODE	R/W	0h	<p>Control Mode Bits</p> <p>Selects the register (CMP/TBPRD or TBPHS) that controls the MEP:</p> <p>0: CMPAHR(8) or TBPRDHR(8) Register controls the edge position (i.e., this is duty or period control mode). (Default on Reset) 1: TBPHSHR(8) Register controls the edge position (i.e., this is phase control mode).</p> <p>Reset type: SYSRSn</p>
1-0	EDGMODE	R/W	0h	<p>Edge Mode Bits</p> <p>Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:</p> <p>00: HRPWM capability is disabled (default on reset) 01: MEP control of rising edge (CMPAHR) 10: MEP control of falling edge (CMPAHR) 11: MEP control of both edges (TBPHSHR or TBPRDHR)</p> <p>Reset type: SYSRSn</p>

### 13.17.2.18 HRPWR Register (Offset = 21h) [Reset = 0000h]

HRPWR is shown in [Figure 13-110](#) and described in [Table 13-41](#).

Return to the [Summary Table](#).

HRPWM Power Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 13-110. HRPWR Register**

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

**Table 13-41. HRPWR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CALPWRON	R/W	0h	MEP Calibration Power Bits (only available on ePWM1) 0: Disables MEP calibration logic in the HRPWM and reduces power consumption. 1: Enables MEP calibration logic Reset type: SYSRSn
14-10	RESERVED	R-0	0h	Reserved
9-6	RESERVED	R/W	0h	Reserved
5	RESERVED	R/W	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	RESERVED	R/W	0h	Reserved
1-0	RESERVED	R/W	0h	Reserved

### 13.17.2.19 HRMSTEP Register (Offset = 26h) [Reset = 0000h]

HRMSTEP is shown in [Figure 13-111](#) and described in [Table 13-42](#).

Return to the [Summary Table](#).

#### HRPWM MEP Step Register

This register is only accessible on EPWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 13-111. HRMSTEP Register**

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

**Table 13-42. 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. Reset type: SYSRSn

### 13.17.2.20 HRCNFG2 Register (Offset = 27h) [Reset = 0000h]

HRCNFG2 is shown in [Figure 13-112](#) and described in [Table 13-43](#).

Return to the [Summary Table](#).

#### HRPWM Configuration 2 Register

This register is only accessible on EPWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 13-112. HRCNFG2 Register**

15	14	13	12	11	10	9	8
RESERVED	RESERVED	RESERVED					
R/W-0h	R-0/W1S-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	

**Table 13-43. HRCNFG2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R/W	0h	Reserved
14	RESERVED	R-0/W1S	0h	Reserved
13-6	RESERVED	R-0	0h	Reserved
5-4	CTLMODEDBFED	R/W	0h	Shadow Mode Bit - selection should match DBCTL[LOADFEDMODE] Selects the time event that loads the DBFEDHR shadow value into the active register. 00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10 Load on either CTR = Zero or CTR = PRD 11 Reserved Reset type: SYSRSn
3-2	CTLMODEDBRED	R/W	0h	Shadow Mode Bit - selection should match DBCTL[LOADREDMODE] Selects the time event that loads the DBREDHR shadow value into the active register. 00 Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 01 Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 10 Load on either CTR = Zero or CTR = PRD 11 Reserved Reset type: SYSRSn
1-0	EDGMODEDB	R/W	0h	Edge Mode Bits Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic: 00 HRPWM capability is disabled (default on reset) 01 MEP control of rising edge (DBREDHR) 10 MEP control of falling edge (DBFEDHR) 11 MEP control of both edges (rising edge of DBREDHR or falling edge of DBFEDHR ) Reset type: SYSRSn

### 13.17.2.21 HRPCTL Register (Offset = 2Dh) [Reset = 0000h]

HRPCTL is shown in [Figure 13-113](#) and described in [Table 13-44](#).

Return to the [Summary Table](#).

High Resolution Period Control Register

Fields in this register related to HRPWM are only applicable on EPWM modules with HRPWM capabilities.

**Figure 13-113. HRPCTL Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	PWMSYNCSELX			RESERVED	TBPHSHRLOA DE	PWMSYNCSEL	HRPE
R-0-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-44. HRPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6-4	PWMSYNCSELX	R/W	0h	Extended selection bits for EPWMSYNCPER 000: EPWMSYNCPER is defined by PWMSYNCSEL - > default condition (compatible with previous EPWM versions) 001: Reserved 010: Reserved 011: Reserved 100: CTR = CMPC, Count direction Up 101: CTR = CMPC, Count direction Down 110: CTR = CMPD, Count direction Up 111: CTR = CMPD, Count direction Down Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2	TBPHSHRLOADE	R/W	0h	TBPHSHR Load Enable This bit allows you to synchronize ePWM modules with a high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event. This allows for multiple ePWM modules operating at the same frequency to be phase aligned with high-resolution. 0: Disables synchronization of high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital compare event: 1: Synchronize the high-resolution phase on a SYNCIN, TBCTL[SWFSYNC] or digital comparator synchronization event. The phase is synchronized using the contents of the high-resolution phase TBPHSHR register. The TBCTL[PHESEN] bit which enables the loading of the TBCTR register with TBPHS register value on a SYNCIN or TBCTL[SWFSYNC] event works independently. However, users need to enable this bit also if they want to control phase in conjunction with the high-resolution period feature. This bit and the TBCTL[PHESEN] bit must be set to 1 when high-resolution period is enabled for up-down count mode even if TBPHSHR = 0x0000. This bit does not need to be set when only high-resolution duty is enabled. Reset type: SYSRSn
1	PWMSYNCSEL	R/W	0h	PWMSYNC Source Select Bit: This bit selects the source for the EPWMSYNCPER signal that goes to the CMPSS and GPDAC: 0 CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 1 CTR = zero: Time-base counter equal to zero (TBCTR = 0x00) Reset type: SYSRSn

**Table 13-44. HRPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	HRPE	R/W	0h	High Resolution Period Enable Bit 0: High resolution period feature disabled. In this mode the ePWM behaves as a Type 4 ePWM. 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. Reset type: SYSRSn

### 13.17.2.22 TRREM Register (Offset = 2Eh) [Reset = 0000h]

TRREM is shown in [Figure 13-114](#) and described in [Table 13-45](#).

Return to the [Summary Table](#).

HRPWM High Resolution Remainder Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 13-114. TRREM Register**

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

**Table 13-45. TRREM Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10-0	TRREM	R/W	0h	<p>HRPWM Remainder Bits: This 11-bit value keeps track of the remainder portion of the HRPWM algorithm calculations. This value keeps track of the remainder portion of the HRPWM hardware calculations.</p> <p>Notes:</p> <ol style="list-style-type: none"> <li>The lower 8-bits of the TRREM register can be automatically initialized with the TBPHSHR value on a SYNCIN or TBCTL[SWFSYNC] event or DC event (if enabled). The user can also write a value with the CPU.</li> <li>Priority of TRREM register updates: Sync (software or hardware) TBPHSHR copied to TRREM : Highest Priority HRPWM Hardware (updates TRREM register): Next priority CPU Write To TRREM Register: Lowest Priority</li> <li>Bit 10 of TRREM register is not used in asymmetrical mode. This bit can be forced to zero. TRREM will be initialized to 0x0 and 0x100 in Up and Up-down modes respectively. Asymmetrical Mode: TRREM[7:0] = TBPHSHR[15:8] TRREM[10,9,8] = 0,0,0 Symmetrical Mode: TRREM[7:0] = TBPHSHR[15:8] TRREM[10,9,8] = 0,0,1 Reset type: SYSRSn</li> </ol>

### 13.17.2.23 GLDCTL Register (Offset = 34h) [Reset = 0000h]

GLDCTL is shown in [Figure 13-115](#) and described in [Table 13-46](#).

Return to the [Summary Table](#).

Global PWM Load Control Register

**Figure 13-115. GLDCTL Register**

15	14	13	12	11	10	9	8
RESERVED			GLDCNT			GLDPRD	
R-0-0h			R-0h			R/W-0h	
7	6	5	4	3	2	1	0
GLDPRD	RESERVED	OSHTMODE	GLDMODE			GLD	
R/W-0h	R-0-0h	R/W-0h	R/W-0h			R/W-0h	

**Table 13-46. GLDCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R-0	0h	Reserved
12-10	GLDCNT	R	0h	Global Load Strobe Counter Register These bits indicate how many selected events have occurred: 000: No events 001: 1 event 010: 2 events 011: 3 events 100: 4 events 101: 5 events 110: 6 events 111: 7 events Reset type: SYSRSn
9-7	GLDPRD	R/W	0h	Global Load Strobe Period Select Register These bits select how many selected events need to occur before a load strobe is generated 000: Disable counter 001: Generate strobe on GLDCNT = 001 (1st event) 010: Generate strobe on GLDCNT = 010 (2nd event) 011: Generate strobe on GLDCNT = 011 (3rd event) 100: Generate strobe on GLDCNT = 011 (4th event) 101: Generate strobe on GLDCNT = 001 (5th event) 110: Generate strobe on GLDCNT = 010 (6th event) 111: Generate strobe on GLDCNT = 011 (7th event) Reset type: SYSRSn
6	RESERVED	R-0	0h	Reserved
5	OSHTMODE	R/W	0h	One Shot Load Mode Control Bit 0: One shot load mode is disabled and shadow to active loading happens continuously on all the chosen load strobes. 1: One shot mode is active. All load strobes are blocked until GLDCTL2[OSHTLD] is written with 1. Note: One Shot mode can only be used with global shadow to active load mode enabled (GLDCTL[GLD]=1) Reset type: SYSRSn

**Table 13-46. 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 GLDCTL2[GFRCLD] write</p> <p>Reset type: SYSRSn</p>
0	GLD	R/W	0h	<p>Global Shadow to Active Load Event Control</p> <p>0: Shadow to active reload for all shadowed registers happens as per the individual reload control bits specified (Compatible with previous EPWM versions).</p> <p>1: When set, all the shadow to active reload events are defined by GLDMODE bits in GLDCTL register. All the shadow registers use same reload pulse from shadow to active reloading. Individual LOADMODE bits are ignored.</p> <p>Reset type: SYSRSn</p>

### 13.17.2.24 GLDCFG Register (Offset = 35h) [Reset = 0000h]

GLDCFG is shown in [Figure 13-116](#) and described in [Table 13-47](#).

Return to the [Summary Table](#).

Global PWM Load Config Register

**Figure 13-116. 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		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		R/W-0h		

**Table 13-47. GLDCFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-11	RESERVED	R-0	0h	Reserved
10	AQCSFRC	R/W	0h	Global load event configuration for AQCSFRC 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
9	AQCTLB_AQCTLB2	R/W	0h	Global load event configuration for AQCTLB_AQCTLB2 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
8	AQCTLA_AQCTLA2	R/W	0h	Global load event configuration for AQCTLA_AQCTLA2 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
7	DBCTL	R/W	0h	Global load event configuration for DBCTL 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
6	DBFED_DBFEDHR	R/W	0h	Global load event configuration for DBFED_DBFEDHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
5	DBRED_DBREDHR	R/W	0h	Global load event configuration for DBRED_DBREDHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn

**Table 13-47. GLDCFG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	CMPD	R/W	0h	Global load event configuration for CMPD 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
3	CMPC	R/W	0h	Global load event configuration for CMPC 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
2	CMPB_CMPBHR	R/W	0h	Global load event configuration for CMPB_CMPBHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
1	CMPA_CMPAHR	R/W	0h	Global load event configuration for CMPA_CMPAHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn
0	TBPRD_TBPRDHR	R/W	0h	Global load event configuration for TBPRD_TBPRDHR 0: Registers use local reload configuration even if GLDCTL(GLD)=1 (reload is compatible with previous EPWMs) 1: Registers use global load configuration if this bit is set and GLDCTL(GLD)=1 Reset type: SYSRSn

### 13.17.2.25 EPWMXLINK Register (Offset = 38h) [Reset = 000XXXXh]

EPWMXLINK is shown in [Figure 13-117](#) and described in [Table 13-48](#).

Return to the [Summary Table](#).

#### EPWMx Link Register

This register controls which EPWMs are linked to other EPWM modules. The default reset value will vary for each module. The reset value will link each EPWM module to itself to prevent unintentional linking of modules.

**Figure 13-117. EPWMXLINK Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GLDCTL2LINK				RESERVED								CMPDLINK			
R/W-0h				R-0-0h								R/W-X			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMPCLINK				CMPBLINK				CMPALINK				TBPRDLINK			
R/W-X				R/W-X				R/W-X				R/W-X			

**Table 13-48. EPWMXLINK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	GLDCTL2LINK	R/W	0h	GLDCTL2 Link Bits 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. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn
27-20	RESERVED	R-0	0h	Reserved
19-16	CMPDLINK	R/W	X	CMPD Link Bits 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. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn
15-12	CMPCLINK	R/W	X	CMPC Link Bits 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. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn
11-8	CMPBLINK	R/W	X	CMPB_CMPBHR Link Bits Writes to the CMPB_CMPBHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPB_CMPBHR registers. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn

**Table 13-48. EPWMXLINK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-4	CMPALINK	R/W	X	CMPA_CMPAHR Link Bits Writes to the CMPA_CMPAHR registers in the ePWM module selected by the following bit selections results in a simultaneous write to the current ePWM module's CMPA_CMPAHR registers. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn
3-0	TBPRDLINK	R/W	X	TBPRD_TBPRDHR Link Bits 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. 0000: ePWM1 0001: ePWM2 ... Up to the last instance of ePWM. All others are reserved. Reset type: SYSRSn

**13.17.2.26 AQCTLA Register (Offset = 40h) [Reset = 0000h]**

 AQCTLA is shown in [Figure 13-118](#) and described in [Table 13-49](#).

 Return to the [Summary Table](#).

Action Qualifier Control Register For Output A

**Figure 13-118. AQCTLA Register**

15	14	13	12	11	10	9	8
RESERVED				CBD		CBU	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-49. AQCTLA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	CBD	R/W	0h	Action When TBCTR = CMPB on Down Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
9-8	CBU	R/W	0h	Action When TBCTR = CMPB on Up Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
7-6	CAD	R/W	0h	Action When TBCTR = CMPA on Down Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 13-49. AQCTLA Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	PRD	R/W	0h	Action When TBCTR = TBPRD Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
1-0	ZRO	R/W	0h	Action When TBCTR = 0 Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxA output low. 10: Set: force EPWMxA output high. 11: Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn

**13.17.2.27 AQCTLA2 Register (Offset = 41h) [Reset = 0000h]**

 AQCTLA2 is shown in [Figure 13-119](#) and described in [Table 13-50](#).

 Return to the [Summary Table](#).

Additional Action Qualifier Control Register For Output A

**Figure 13-119. AQCTLA2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2D		T2U		T1D		T1U	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-50. 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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn

### 13.17.2.28 AQCTLB Register (Offset = 42h) [Reset = 0000h]

AQCTLB is shown in [Figure 13-120](#) and described in [Table 13-51](#).

Return to the [Summary Table](#).

Action Qualifier Control Register For Output B

**Figure 13-120. AQCTLB Register**

15	14	13	12	11	10	9	8
RESERVED				CBD		CBU	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-51. AQCTLB Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	CBD	R/W	0h	Action When TBCTR = CMPB on Down Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
9-8	CBU	R/W	0h	Action When TBCTR = CMPB on Up Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
7-6	CAD	R/W	0h	Action When TBCTR = CMPA on Down Count Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 13-51. AQCTLB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-2	PRD	R/W	0h	Action When TBCTR = TBPRD Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn
1-0	ZRO	R/W	0h	Action When TBCTR = 0 Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 00: Do nothing (action disabled) 01: Clear: force EPWMxB output low. 10: Set: force EPWMxB output high. 11: Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low. Reset type: SYSRSn

### 13.17.2.29 AQCTLB2 Register (Offset = 43h) [Reset = 0000h]

AQCTLB2 is shown in [Figure 13-121](#) and described in [Table 13-52](#).

Return to the [Summary Table](#).

Additional Action Qualifier Control Register For Output B

**Figure 13-121. AQCTLB2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
T2D		T2U		T1D		T1U	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-52. 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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn

### 13.17.2.30 AQSFRFC Register (Offset = 47h) [Reset = 0000h]

AQSFRFC is shown in [Figure 13-122](#) and described in [Table 13-53](#).

Return to the [Summary Table](#).

Action Qualifier Software Force Register

**Figure 13-122. 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	
R/W-0h		R-0/W1S-0h		R/W-0h		R/W-0h	

**Table 13-53. AQSFRFC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7-6	RLDCSF	R/W	0h	AQSFRFC Active Register Reload From Shadow Options 00: Load on time-base counter equals zero 01: Load on time-base counter equals period 10: Load on time-base counter equals zero or counter equals period 11: Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register). Reset type: SYSRSn
5	OTSFB	R-0/W1S	0h	One-Time Software Forced Event on Output B 0: Writing a 0 (zero) has no effect. Always reads back a 0. This bit is auto cleared once a write to this register is complete (i.e., a forced event is initiated.). This is a one-shot forced event. It can be overridden by another subsequent event on output B. 1: Initiates a single software forced event Reset type: SYSRSn
4-3	ACTSFB	R/W	0h	Action When One-Time Software Force B is Invoked 00: Does nothing (action disabled) 01: Clear (low) 10: Set (high) 11: Toggle (Low -> High, High -> Low) Note: This action is not qualified by counter direction (CNT_dir) Reset type: SYSRSn
2	OTSFA	R-0/W1S	0h	One-Time Software Forced Event on Output A 0: Writing a 0 (zero) has no effect. Always reads back a 0. This bit is auto cleared once a write to this register is complete ( i.e., a forced event is initiated). This is a one-shot forced event. It can be overridden by another subsequent event on output A. 1: Initiates a single software forced event Reset type: SYSRSn
1-0	ACTSFA	R/W	0h	Action When One-Time Software Force A Is Invoked 00: Does nothing (action disabled) 01: Clear (low) 10: Set (high) 11: Toggle (Low -> High, High -> Low) Note: This action is not qualified by counter direction (CNT_dir) Reset type: SYSRSn

### 13.17.2.31 AQCSFRC Register (Offset = 49h) [Reset = 0000h]

AQCSFRC is shown in [Figure 13-123](#) and described in [Table 13-54](#).

Return to the [Summary Table](#).

Action Qualifier Continuous S/W Force Register

**Figure 13-123. AQCSFRC Register**

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

**Table 13-54. 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: Software forcing is disabled and has no effect 01: Forces a continuous low on output B 10: Forces a continuous high on output B 11: Software forcing is disabled and has no effect Reset type: SYSRSn
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: Software forcing is disabled and has no effect 01: Forces a continuous low on output A 10: Forces a continuous high on output A 11: Software forcing is disabled and has no effect Reset type: SYSRSn

### 13.17.2.32 DBREDHR Register (Offset = 50h) [Reset = 0000h]

DBREDHR is shown in [Figure 13-124](#) and described in [Table 13-55](#).

Return to the [Summary Table](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 13-124. 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

**Table 13-55. DBREDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	DBREDHR	R/W	0h	Dead Band Rising Edge Delay High Resolution Bits Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

### 13.17.2.33 DBRED Register (Offset = 51h) [Reset = 0000h]

DBRED is shown in [Figure 13-125](#) and described in [Table 13-56](#).

Return to the [Summary Table](#).

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 13-125. DBRED Register**

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

**Table 13-56. DBRED Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13-0	DBRED	R/W	0h	Rising edge delay value Reset type: SYSRSn

### 13.17.2.34 DBFEDHR Register (Offset = 52h) [Reset = 0000h]

DBFEDHR is shown in [Figure 13-126](#) and described in [Table 13-57](#).

Return to the [Summary Table](#).

Dead-Band Generator Falling Edge Delay High Resolution Register

**Figure 13-126. 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

**Table 13-57. DBFEDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	DBFEDHR	R/W	0h	Dead Band Falling Edge Delay High Resolution Bits Reset type: SYSRSn
8	RESERVED	R	0h	Reserved
7-1	RESERVED	R	0h	Reserved
0	RESERVED	R	0h	Reserved

### 13.17.2.35 DBFED Register (Offset = 53h) [Reset = 0000h]

DBFED is shown in [Figure 13-127](#) and described in [Table 13-58](#).

Return to the [Summary Table](#).

Dead-Band Generator Falling Edge Delay Count Register

**Figure 13-127. DBFED Register**

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

**Table 13-58. DBFED Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13-0	DBFED	R/W	0h	Falling Edge Delay Count 14-bit counter Reset type: SYSRSn

### 13.17.2.36 TBPHS Register (Offset = 60h) [Reset = 0000000h]

TBPHS is shown in [Figure 13-128](#) and described in [Table 13-59](#).

Return to the [Summary Table](#).

Time Base Phase High

**Figure 13-128. TBPHS Register**

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

**Table 13-59. TBPHS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	TBPHS	R/W	0h	Phase Offset Register These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal. - If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. - If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization. Reset type: SYSRSn
15-0	TBPHSHR	R/W	0h	Phase Offset (High Resolution) Register. TBPHSHR must not be used. Instead TRREM (HRPWM remainder register) must be used to mimic the functionality of TBPHSHR. The lower 8 bits in this register are ignored - writes are ignored and reads return zero Reset type: SYSRSn

**13.17.2.37 TBPRDHR Register (Offset = 62h) [Reset = 0000h]**

TBPRDHR is shown in [Figure 13-129](#) and described in [Table 13-60](#).

Return to the [Summary Table](#).

Time Base Period High Resolution Register

**Figure 13-129. TBPRDHR Register**

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

**Table 13-60. TBPRDHR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBPRDHR	R/W	0h	Period High Resolution Bits The upper 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDL] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled. This register is only available with ePWM modules which support high-resolution period control. The lower 8 bits in this register are ignored - writes are ignored and reads return zero Reset type: SYSRSn

### 13.17.2.38 TBPRD Register (Offset = 63h) [Reset = 0000h]

TBPRD is shown in [Figure 13-130](#) and described in [Table 13-61](#).

Return to the [Summary Table](#).

Time Base Period Register

**Figure 13-130. TBPRD Register**

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

**Table 13-61. TBPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TBPRD	R/W	0h	Time Base Period Register 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. - 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. Reset type: SYSRSn

### 13.17.2.39 CMPA Register (Offset = 6Ah) [Reset = 0000000h]

CMPA is shown in [Figure 13-131](#) and described in [Table 13-62](#).

Return to the [Summary Table](#).

Counter Compare A Register

**Figure 13-131. CMPA Register**

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

**Table 13-62. CMPA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	CMPA	R/W	0h	<p>Compare A Register</p> <p>The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a 'time-base counter equal to counter compare A' event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>- Do nothing</li> <li>- Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>- Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>- Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register.</li> <li>- Before a write, the CMPCTL[SHDWFULL] bit can be read to determine if the shadow register is currently full.</li> <li>- If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>
15-0	CMPAHR	R/W	0h	<p>Compare A HRPWM Extension Register</p> <p>The UPPER 8-bits contain the high-resolution portion (most significant 8-bits) of the counter-compare A value. CMPA:CMPAHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPA register.</p> <p>The lower 8 bits in this register are ignored</p> <p>Reset type: SYSRSn</p>

### 13.17.2.40 CMPB Register (Offset = 6Ch) [Reset = 0000000h]

CMPB is shown in [Figure 13-132](#) and described in [Table 13-63](#).

Return to the [Summary Table](#).

Compare B Register

**Figure 13-132. CMPB Register**

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

**Table 13-63. CMPB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	CMPB	R/W	0h	<p>Compare B Register</p> <p>The value in the active CMPB register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a 'time-base counter equal to counter compare B' event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>- Do nothing</li> <li>the event is ignored.</li> <li>- Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>- Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>- Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register.</li> <li>- Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full.</li> <li>- If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>
15-0	CMPBHR	R/W	0h	<p>Compare B High Resolution Bits</p> <p>The lower 8 bits in this register are ignored</p> <p>Reset type: SYSRSn</p>

### 13.17.2.41 CMPC Register (Offset = 6Fh) [Reset = 0000h]

CMPC is shown in [Figure 13-133](#) and described in [Table 13-64](#).

Return to the [Summary Table](#).

Counter Compare C Register

LINK feature access should always be 16-bit

**Figure 13-133. CMPC Register**

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

**Table 13-64. CMPC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CMPC	R/W	0h	<p>Compare C Register</p> <p>The value in the active CMPC register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a 'time-base counter equal to counter compare C' event.</p> <p>Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>- If CMPCTL2[SHDWCMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL2[LOADCMODE] bit field determines which event will load the active register from the shadow register:</li> <li>- If CMPCTL2[SHDWCMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register that is, the register actively controlling the hardware.</li> <li>- In either mode, the active and shadow registers share the same memory map address.</li> </ul> <p>Reset type: SYSRSn</p>

### 13.17.2.42 CMPD Register (Offset = 71h) [Reset = 0000h]

CMPD is shown in [Figure 13-134](#) and described in [Table 13-65](#).

Return to the [Summary Table](#).

Counter Compare D Register

LINK feature access should always be 16-bit

**Figure 13-134. CMPD Register**

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

**Table 13-65. CMPD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	CMPD	R/W	0h	Compare D Register 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. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWDMODE] bit. By default this register is shadowed. - 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. Reset type: SYSRSn

### 13.17.2.43 GLDCTL2 Register (Offset = 74h) [Reset = 0000h]

GLDCTL2 is shown in [Figure 13-135](#) and described in [Table 13-66](#).

Return to the [Summary Table](#).

Global PWM Load Control Register 2

**Figure 13-135. GLDCTL2 Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED						GFRCLD	OSHTLD
R-0-0h						R-0/W1S-0h	R-0/W1S-0h

**Table 13-66. GLDCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R-0	0h	Reserved
1	GFRCLD	R-0/W1S	0h	Force Load Event in One Shot Mode 0: Writing of 0 will be ignored. Always reads back a 0. 1: Force one load event at the input of the event pre-scale counter. This bit is intended to be used for testing and/or software force loading of the events in global load mode. Reset type: SYSRSn
0	OSHTLD	R-0/W1S	0h	Enable Reload Event in One Shot Mode 0: Writing of 0 will be ignored. Always reads back a 0. 1: Turns the one shot latch condition ON. Upon occurrence of a chosen load strobe, one shadow to active reload occurs and the latch will be cleared. Hence writing 1 to this bit would allow one load strobe event to pass through and block further strobe events. Reset type: SYSRSn

### 13.17.2.44 SWVDELVAL Register (Offset = 77h) [Reset = 0000h]

SWVDELVAL is shown in [Figure 13-136](#) and described in [Table 13-67](#).

Return to the [Summary Table](#).

Software Valley Mode Delay Register

**Figure 13-136. SWVDELVAL Register**

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

**Table 13-67. SWVDELVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SWVDELVAL	R/W	0h	Software Valley Delay Value Register This register can be optionally used define offset value for the hardware calculated delay HWDELAYVAL as defined in VCAPCTL[VDELAYDIV] bits. Reset type: SYSRSn

### 13.17.2.45 TZSEL Register (Offset = 80h) [Reset = 0000h]

TZSEL is shown in [Figure 13-137](#) and described in [Table 13-68](#).

Return to the [Summary Table](#).

Trip Zone Select Register

**Figure 13-137. TZSEL Register**

15	14	13	12	11	10	9	8
DCBEVT1	DCAEVT1	OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-68. TZSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Select 0: Disable DCBEVT1 as one-shot-trip source for this ePWM module. 1: Enable DCBEVT1 as one-shot-trip source for this ePWM module. Reset type: SYSRSn
14	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Select 0: Disable DCAEVT1 as one-shot-trip source for this ePWM module. 1: Enable DCAEVT1 as one-shot-trip source for this ePWM module. Reset type: SYSRSn
13	OSHT6	R/W	0h	Trip-zone 6 (TZ6) Select 0: Disable TZ6 as a one-shot trip source for this ePWM module 1: Enable TZ6 as a one-shot trip source for this ePWM module Reset type: SYSRSn
12	OSHT5	R/W	0h	Trip-zone 5 (TZ5) Select 0: Disable TZ5 as a one-shot trip source for this ePWM module 1: Enable TZ5 as a one-shot trip source for this ePWM module Reset type: SYSRSn
11	OSHT4	R/W	0h	Trip-zone 4 (TZ4) Select 0: Disable TZ4 as a one-shot trip source for this ePWM module 1: Enable TZ4 as a one-shot trip source for this ePWM module Reset type: SYSRSn
10	OSHT3	R/W	0h	Trip-zone 3 (TZ3) Select 0: Disable TZ3 as a one-shot trip source for this ePWM module 1: Enable TZ3 as a one-shot trip source for this ePWM module Reset type: SYSRSn
9	OSHT2	R/W	0h	Trip-zone 2 (TZ2) Select 0: Disable TZ2 as a one-shot trip source for this ePWM module 1: Enable TZ2 as a one-shot trip source for this ePWM module Reset type: SYSRSn
8	OSHT1	R/W	0h	Trip-zone 1 (TZ1) Select 0: Disable TZ1 as a one-shot trip source for this ePWM module 1: Enable TZ1 as a one-shot trip source for this ePWM module Reset type: SYSRSn
7	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Select 0: Disable DCBEVT2 as a CBC trip source for this ePWM module 1: Enable DCBEVT2 as a CBC trip source for this ePWM module Reset type: SYSRSn

**Table 13-68. TZSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Select 0: Disable DCAEVT2 as a CBC trip source for this ePWM module 1: Enable DCAEVT2 as a CBC trip source for this ePWM module Reset type: SYSRSn
5	CBC6	R/W	0h	Trip-zone 6 (TZ6) Select 0: Disable TZ6 as a CBC trip source for this ePWM module 1: Enable TZ6 as a CBC trip source for this ePWM module Reset type: SYSRSn
4	CBC5	R/W	0h	Trip-zone 5 (TZ5) Select 0: Disable TZ5 as a CBC trip source for this ePWM module 1: Enable TZ5 as a CBC trip source for this ePWM module Reset type: SYSRSn
3	CBC4	R/W	0h	Trip-zone 4 (TZ4) Select 0: Disable TZ4 as a CBC trip source for this ePWM module 1: Enable TZ4 as a CBC trip source for this ePWM module Reset type: SYSRSn
2	CBC3	R/W	0h	Trip-zone 3 (TZ3) Select 0: Disable TZ3 as a CBC trip source for this ePWM module 1: Enable TZ3 as a CBC trip source for this ePWM module Reset type: SYSRSn
1	CBC2	R/W	0h	Trip-zone 2 (TZ2) Select 0: Disable TZ2 as a CBC trip source for this ePWM module 1: Enable TZ2 as a CBC trip source for this ePWM module Reset type: SYSRSn
0	CBC1	R/W	0h	Trip-zone 1 (TZ1) Select 0: Disable TZ1 as a CBC trip source for this ePWM module 1: Enable TZ1 as a CBC trip source for this ePWM module Reset type: SYSRSn

### 13.17.2.46 TZDCSEL Register (Offset = 82h) [Reset = 0000h]

TZDCSEL is shown in [Figure 13-138](#) and described in [Table 13-69](#).

Return to the [Summary Table](#).

Trip Zone Digital Comparator Select Register

**Figure 13-138. TZDCSEL Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2			DCBEVT1
R-0-0h				R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
DCBEVT1		DCAEVT2			DCAEVT1		
R/W-0h		R/W-0h			R/W-0h		

**Table 13-69. TZDCSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-9	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Selection 000: Event disabled 001: DCBH = low, DCBL = don't care 010: DCBH = high, DCBL = don't care 011: DCBL = low, DCBH = don't care 100: DCBL = high, DCBH = don't care 101: DCBL = high, DCBH = low 110: Reserved 111: Reserved Reset type: SYSRSn
8-6	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Selection 000: Event disabled 001: DCBH = low, DCBL = don't care 010: DCBH = high, DCBL = don't care 011: DCBL = low, DCBH = don't care 100: DCBL = high, DCBH = don't care 101: DCBL = high, DCBH = low 110: Reserved 111: Reserved Reset type: SYSRSn
5-3	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Selection 000: Event disabled 001: DCAH = low, DCAL = don't care 010: DCAH = high, DCAL = don't care 011: DCAL = low, DCAH = don't care 100: DCAL = high, DCAH = don't care 101: DCAL = high, DCAH = low 110: Reserved 111: Reserved Reset type: SYSRSn
2-0	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Selection 000: Event disabled 001: DCAH = low, DCAL = don't care 010: DCAH = high, DCAL = don't care 011: DCAL = low, DCAH = don't care 100: DCAL = high, DCAH = don't care 101: DCAL = high, DCAH = low 110: Reserved 111: Reserved Reset type: SYSRSn

**13.17.2.47 TZCTL Register (Offset = 84h) [Reset = 0000h]**

 TZCTL is shown in [Figure 13-139](#) and described in [Table 13-70](#).

 Return to the [Summary Table](#).

Trip Zone Control Register

**Figure 13-139. TZCTL Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2		DCBEVT1	
R-0-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
DCAEVT2		DCAEVT1		TZB		TZA	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 13-70. TZCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-10	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Action On EPWMxB 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state. 10: Force EPWMxB to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
9-8	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Action On EPWMxB 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state. 10: Force EPWMxB to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
7-6	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA 00: High-impedance (EPWMxA = High-impedance state) 01: Force EPWMxA to a high state. 10: Force EPWMxA to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
5-4	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA 00: High-impedance (EPWMxA = High-impedance state) 01: Force EPWMxA to a high state. 10: Force EPWMxA to a low state. 11: Do Nothing, trip action is disabled Reset type: SYSRSn
3-2	TZB	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxB When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register. 00: High-impedance (EPWMxB = High-impedance state) 01: Force EPWMxB to a high state 10: Force EPWMxB to a low state 11: Do nothing, no action is taken on EPWMxB. Reset type: SYSRSn

**Table 13-70. TZCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	TZA	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA 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. 00: High-impedance (EPWMxA = High-impedance state) 01: Force EPWMxA to a high state 10: Force EPWMxA to a low state 11: Do nothing, no action is taken on EPWMxA. Reset type: SYSRSn

### 13.17.2.48 TZCTL2 Register (Offset = 85h) [Reset = 0000h]

TZCTL2 is shown in [Figure 13-140](#) and described in [Table 13-71](#).

Return to the [Summary Table](#).

Additional Trip Zone Control Register

**Figure 13-140. TZCTL2 Register**

15	14	13	12	11	10	9	8
ETZE	RESERVED			TZBD			TZBU
R/W-0h	R-0-0h			R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
TZBU		TZAD			TZAU		
R/W-0h		R/W-0h			R/W-0h		

**Table 13-71. 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 Reset type: SYSRSn
14-12	RESERVED	R-0	0h	Reserved
11-9	TZBD	R/W	0h	TZ1 to TZ6 Trip Action On EPWMxB while Count direction is DOWN 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	TZBU	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxB while Count direction is UP 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
5-3	TZAD	R/W	0h	TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA while Count direction is DOWN 000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 13-71. 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 Reset type: SYSRSn

### 13.17.2.49 TZCTLDCA Register (Offset = 86h) [Reset = 0000h]

TZCTLDCA is shown in [Figure 13-141](#) and described in [Table 13-72](#).

Return to the [Summary Table](#).

Trip Zone Control Register Digital Compare A

**Figure 13-141. TZCTLDCA Register**

15	14	13	12	11	10	9	8
RESERVED				DCAEVT2D			DCAEVT2U
R-0-0h				R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
DCAEVT2U		DCAEVT1D			DCAEVT1U		
R/W-0h		R/W-0h			R/W-0h		

**Table 13-72. TZCTLDCA Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R-0	0h	Reserved
11-9	DCAEVT2D	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA while Count direction is DOWN 000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	DCAEVT2U	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA while Count direction is UP 000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
5-3	DCAEVT1D	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA while Count direction is DOWN 000: HiZ (EPWMxA = HiZ state) 001: Forced Hi (EPWMxA = High state) 010: Forced Lo (EPWMxA = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 13-72. 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 Reset type: SYSRSn

### 13.17.2.50 TZCTLDCB Register (Offset = 87h) [Reset = 0000h]

TZCTLDCB is shown in [Figure 13-142](#) and described in [Table 13-73](#).

Return to the [Summary Table](#).

Trip Zone Control Register Digital Compare B

**Figure 13-142. TZCTLDCB Register**

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2D			DCBEVT2U
R-0-0h				R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
DCBEVT2U		DCBEVT1D			DCBEVT1U		
R/W-0h		R/W-0h			R/W-0h		

**Table 13-73. 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 B Event 2 Action On EPWMxB while Count direction is DOWN 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
8-6	DCBEVT2U	R/W	0h	Digital Compare Output B Event 2 Action On EPWMxB while Count direction is UP 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn
5-3	DCBEVT1D	R/W	0h	Digital Compare Output B Event 1 Action On EPWMxB while Count direction is DOWN 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

**Table 13-73. TZCTLDCB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	DCBEVT1U	R/W	0h	Digital Compare Output B Event 1 Action On EPWMxB while Count direction is UP 000: HiZ (EPWMxB = HiZ state) 001: Forced Hi (EPWMxB = High state) 010: Forced Lo (EPWMxB = Lo state) 011: Toggle (Low -> High, High -> Low) 100: Reserved 101: Reserved 110: Reserved 111: Do Nothing, trip action is disabled Reset type: SYSRSn

### 13.17.2.51 TZEINT Register (Offset = 8Dh) [Reset = 0000h]

TZEINT is shown in [Figure 13-143](#) and described in [Table 13-74](#).

Return to the [Summary Table](#).

Trip Zone Enable Interrupt Register

**Figure 13-143. TZEINT Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0-0h

**Table 13-74. TZEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
5	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
4	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
3	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Interrupt Enable 0: Disabled 1: Enabled Reset type: SYSRSn
2	OST	R/W	0h	Trip-zone One-Shot Interrupt Enable 0: Disable one-shot interrupt generation 1: Enable Interrupt generation a one-shot trip event will cause a EPWMx_TZINT PIE interrupt. Reset type: SYSRSn
1	CBC	R/W	0h	Trip-zone Cycle-by-Cycle Interrupt Enable 0: Disable cycle-by-cycle interrupt generation. 1: Enable interrupt generation a cycle-by-cycle trip event will cause an EPWMx_TZINT PIE interrupt. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 13.17.2.52 TZFLG Register (Offset = 93h) [Reset = 0000h]

TZFLG is shown in [Figure 13-144](#) and described in [Table 13-75](#).

Return to the [Summary Table](#).

Trip Zone Flag Register

**Figure 13-144. TZFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 13-75. TZFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R	0h	Latched Status Flag for Digital Compare Output B Event 2 0: Indicates no trip event has occurred on DCBEVT2 1: Indicates a trip event has occurred for the event defined for DCBEVT2 Reset type: SYSRSn
5	DCBEVT1	R	0h	Latched Status Flag for Digital Compare Output B Event 1 0: Indicates no trip event has occurred on DCBEVT1 1: Indicates a trip event has occurred for the event defined for DCBEVT1 Reset type: SYSRSn
4	DCAEVT2	R	0h	Latched Status Flag for Digital Compare Output A Event 2 0: Indicates no trip event has occurred on DCAEVT2 1: Indicates a trip event has occurred for the event defined for DCAEVT2 Reset type: SYSRSn
3	DCAEVT1	R	0h	Latched Status Flag for Digital Compare Output A Event 1 0: Indicates no trip event has occurred on DCAEVT1 1: Indicates a trip event has occurred for the event defined for DCAEVT1 Reset type: SYSRSn
2	OST	R	0h	Latched Status Flag for A One-Shot Trip Event 0: No one-shot trip event has occurred. 1: Indicates a trip event has occurred on a pin selected as a one-shot trip source. This bit is cleared by writing the appropriate value to the TZCLR register. Reset type: SYSRSn

**Table 13-75. TZFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: SYSRSn
0	INT	R	0h	Latched Trip Interrupt Status Flag 0: Indicates no interrupt has been generated. 1: Indicates an EPWMx_TZINT PIE interrupt was generated because of a trip condition. No further EPWMx_TZINT PIE interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts. This bit is cleared by writing the appropriate value to the TZCLR register. Reset type: SYSRSn

### 13.17.2.53 TZCBCFLG Register (Offset = 94h) [Reset = 0000h]

TZCBCFLG is shown in [Figure 13-145](#) and described in [Table 13-76](#).

Return to the [Summary Table](#).

Trip Zone CBC Flag Register

**Figure 13-145. TZCBCFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 13-76. TZCBCFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT2	R	0h	Latched Status Flag for Digital Compare B Output Event 2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCBEVT2. 1: Reading a 1 indicates a trip has occurred on the DCBEVT2 selected event. Reset type: SYSRSn
6	DCAEVT2	R	0h	Latched Status Flag for Digital Compare A Output Event 2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCAEVT2. 1: Reading a 1 indicates a trip has occurred on the DCAEVT2 selected event. Reset type: SYSRSn
5	CBC6	R	0h	Latched Status Flag for CBC6 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC6. 1: Reading a 1 indicates a trip has occurred on the CBC6 selected event. Reset type: SYSRSn
4	CBC5	R	0h	Latched Status Flag for CBC5 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC5. 1: Reading a 1 indicates a trip has occurred on the CBC5 selected event. Reset type: SYSRSn
3	CBC4	R	0h	Latched Status Flag for CBC4 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC4. 1: Reading a 1 indicates a trip has occurred on the CBC4 selected event. Reset type: SYSRSn
2	CBC3	R	0h	Latched Status Flag for CBC3 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC3. 1: Reading a 1 indicates a trip has occurred on the CBC3 selected event. Reset type: SYSRSn
1	CBC2	R	0h	Latched Status Flag for CBC2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC2. 1: Reading a 1 indicates a trip has occurred on the CBC2 selected event. Reset type: SYSRSn

**Table 13-76. TZCBCFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	CBC1	R	0h	Latched Status Flag for CBC1 Trip Latch 0: Reading a 0 indicates that no trip has occurred on CBC1. 1: Reading a 1 indicates a trip has occurred on the CBC1 selected event. Reset type: SYSRSn

### 13.17.2.54 TZOSTFLG Register (Offset = 95h) [Reset = 0000h]

TZOSTFLG is shown in [Figure 13-146](#) and described in [Table 13-77](#).

Return to the [Summary Table](#).

Trip Zone OST Flag Register

**Figure 13-146. TZOSTFLG Register**

15		14		13		12		11		10		9		8	
RESERVED															
R-0-0h															
7		6		5		4		3		2		1		0	
DCBEVT1		DCAEVT1		OST6		OST5		OST4		OST3		OST2		OST1	
R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h		R-0h	

**Table 13-77. TZOSTFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT1	R	0h	Latched Status Flag for Digital Compare B Output Event 1 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCBEVT1. 1: Reading a 1 indicates a trip has occurred on the DCBEVT1 selected event. Reset type: SYSRSn
6	DCAEVT1	R	0h	Latched Status Flag for Digital Compare A Output Event 1 Trip Latch 0: Reading a 0 indicates that no trip has occurred on DCAEVT1. 1: Reading a 1 indicates a trip has occurred on the DCAEVT1 selected event. Reset type: SYSRSn
5	OST6	R	0h	Latched Status Flag for OST6 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST6. 1: Reading a 1 indicates a trip has occurred on the OST6 selected event. Reset type: SYSRSn
4	OST5	R	0h	Latched Status Flag for OST5 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST5. 1: Reading a 1 indicates a trip has occurred on the OST5 selected event. Reset type: SYSRSn
3	OST4	R	0h	Latched Status Flag for OST4 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST4. 1: Reading a 1 indicates a trip has occurred on the OST4 selected event. Reset type: SYSRSn
2	OST3	R	0h	Latched Status Flag for OST3 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST3. 1: Reading a 1 indicates a trip has occurred on the OST3 selected event. Reset type: SYSRSn
1	OST2	R	0h	Latched Status Flag for OST2 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST2. 1: Reading a 1 indicates a trip has occurred on the OST2 selected event. Reset type: SYSRSn

**Table 13-77. TZOSTFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	OST1	R	0h	Latched Status Flag for OST1 Trip Latch 0: Reading a 0 indicates that no trip has occurred on OST1. 1: Reading a 1 indicates a trip has occurred on the OST1 selected event. Reset type: SYSRSn

### 13.17.2.55 TZCLR Register (Offset = 97h) [Reset = 0000h]

TZCLR is shown in [Figure 13-147](#) and described in [Table 13-78](#).

Return to the [Summary Table](#).

Trip Zone Clear Register

**Figure 13-147. TZCLR Register**

15	14	13	12	11	10	9	8
CBCPULSE		RESERVED					
R/W-0h		R-0-0h					
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0-0h	R-0/W1S-0h						

**Table 13-78. TZCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	CBCPULSE	R/W	0h	Clear Pulse for Cycle-By-Cycle (CBC) Trip Latch This bit field determines which pulse clears the CBC trip latch. 00: CTR = zero pulse clears CBC trip latch. (Same as legacy designs.) 01: CTR = PRD pulse clears CBC trip latch. 10: CTR = zero or CTR = PRD pulse clears CBC trip latch. 11: CBC trip latch is not cleared Reset type: SYSRSn
13-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 2 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCBEVT2 event trip condition. Reset type: SYSRSn
5	DCBEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 1 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCBEVT1 event trip condition. Reset type: SYSRSn
4	DCAEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 2 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCAEVT2 event trip condition. Reset type: SYSRSn
3	DCAEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 1 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 clears the DCAEVT1 event trip condition. Reset type: SYSRSn
2	OST	R-0/W1S	0h	Clear Flag for One-Shot Trip (OST) Latch 0: Has no effect. Always reads back a 0. 1: Clears this Trip (set) condition. Reset type: SYSRSn
1	CBC	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC) Trip Latch 0: Has no effect. Always reads back a 0. 1: Clears this Trip (set) condition. Reset type: SYSRSn

**Table 13-78. TZCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT	R-0/W1S	0h	Global Interrupt Clear Flag 0: Has no effect. Always reads back a 0. 1: Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]). NOTE: No further EPWMx_TZINT PIE interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts. Reset type: SYSRSn

### 13.17.2.56 TZCBCCLR Register (Offset = 98h) [Reset = 0000h]

TZCBCCLR is shown in [Figure 13-148](#) and described in [Table 13-79](#).

Return to the [Summary Table](#).

Trip Zone CBC Clear Register

**Figure 13-148. TZCBCCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0/W1S-0h							

**Table 13-79. TZCBCCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 2 selected for CBC 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[DCBEVT2] bit. Reset type: SYSRSn
6	DCAEVT2	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 2 selected for CBC 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[DCAEVT2] bit. Reset type: SYSRSn
5	CBC6	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC6) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC6] bit. Reset type: SYSRSn
4	CBC5	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC5) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC5] bit. Reset type: SYSRSn
3	CBC4	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC4) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC4] bit. Reset type: SYSRSn
2	CBC3	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC3) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC3] bit. Reset type: SYSRSn
1	CBC2	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC2) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC2] bit. Reset type: SYSRSn
0	CBC1	R-0/W1S	0h	Clear Flag for Cycle-By-Cycle (CBC1) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZCBCFLG[CBC1] bit. Reset type: SYSRSn

### 13.17.2.57 TZOSTCLR Register (Offset = 99h) [Reset = 0000h]

TZOSTCLR is shown in [Figure 13-149](#) and described in [Table 13-80](#).

Return to the [Summary Table](#).

Trip Zone OST Clear Register

**Figure 13-149. TZOSTCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
DCBEVT1	DCAEVT1	OST6	OST5	OST4	OST3	OST2	OST1
R-0/W1S-0h							

**Table 13-80. TZOSTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R-0	0h	Reserved
7	DCBEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output B Event 1 selected for OST 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[DCBEVT1] bit. Reset type: SYSRSn
6	DCAEVT1	R-0/W1S	0h	Clear Flag for Digital Compare Output A Event 1 selected for OST 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[DCAEVT1] bit. Reset type: SYSRSn
5	OST6	R-0/W1S	0h	Clear Flag for Oneshot (OST6) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST6] bit. Reset type: SYSRSn
4	OST5	R-0/W1S	0h	Clear Flag for Oneshot (OST5) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST5] bit. Reset type: SYSRSn
3	OST4	R-0/W1S	0h	Clear Flag for Oneshot (OST4) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST4] bit. Reset type: SYSRSn
2	OST3	R-0/W1S	0h	Clear Flag for Oneshot (OST3) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST3] bit. Reset type: SYSRSn
1	OST2	R-0/W1S	0h	Clear Flag for Oneshot (OST2) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST2] bit. Reset type: SYSRSn
0	OST1	R-0/W1S	0h	Clear Flag for Oneshot (OST1) Trip Latch 0: Writing a 0 has no effect. 1: Writing a 1 will clear the TZOSTFLG[OST1] bit. Reset type: SYSRSn

### 13.17.2.58 TZFRC Register (Offset = 9Bh) [Reset = 0000h]

TZFRC is shown in [Figure 13-150](#) and described in [Table 13-81](#).

Return to the [Summary Table](#).

Trip Zone Force Register

**Figure 13-150. TZFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0-0h

**Table 13-81. TZFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R-0	0h	Reserved
6	DCBEVT2	R-0/W1S	0h	Force Flag for Digital Compare Output B Event 2 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCBEVT2 event trip condition and sets the TZFLG[DCBEVT2] bit. Reset type: SYSRSn
5	DCBEVT1	R-0/W1S	0h	Force Flag for Digital Compare Output B Event 1 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCBEVT1 event trip condition and sets the TZFLG[DCBEVT1] bit. Reset type: SYSRSn
4	DCAEVT2	R-0/W1S	0h	Force Flag for Digital Compare Output A Event 2 0: Writing 0 has no effect. This bit always reads back 0. 1: Writing 1 forces the DCAEVT2 event trip condition and sets the TZFLG[DCAEVT2] bit. Reset type: SYSRSn
3	DCAEVT1	R-0/W1S	0h	Force Flag for Digital Compare Output A Event 1 0: Writing 0 has no effect. This bit always reads back 0 1: Writing 1 forces the DCAEVT1 event trip condition and sets the TZFLG[DCAEVT1] bit. Reset type: SYSRSn
2	OST	R-0/W1S	0h	Force a One-Shot Trip Event via Software 0: Writing of 0 is ignored. Always reads back a 0. 1: Forces a one-shot trip event and sets the TZFLG[OST] bit. Reset type: SYSRSn
1	CBC	R-0/W1S	0h	Force a Cycle-by-Cycle Trip Event via Software 0: Writing of 0 is ignored. Always reads back a 0. 1: Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit. Reset type: SYSRSn
0	RESERVED	R-0	0h	Reserved

### 13.17.2.59 ETSEL Register (Offset = A4h) [Reset = 0000h]

ETSEL is shown in [Figure 13-151](#) and described in [Table 13-82](#).

Return to the [Summary Table](#).

Event Trigger Selection Register

**Figure 13-151. ETSEL Register**

15	14	13	12	11	10	9	8
SOCBEN	SOCBSEL			SOCAEN	SOCASEL		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
RESERVED	INTSELCMP	SOCBSELCMP	SOCASELCMP	INTEN	INTSEL		
R-0-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

**Table 13-82. ETSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOCBEN	R/W	0h	Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse 0: Disable EPWMxSOCB. 1: Enable EPWMxSOCB pulse. Reset type: SYSRSn
14-12	SOCBSEL	R/W	0h	EPWMxSOCB Selection Options These bits determine when a EPWMxSOCB pulse will be generated. 000: Enable DCBEVT1.soc event 001: Enable event time-base counter equal to zero. (TBCTR = 0x00) 010: Enable event time-base counter equal to period (TBCTR = TBPRD) 011: Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode. 100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing 101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing 110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing 111: Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by SOCBSELCMP bit. Reset type: SYSRSn
11	SOCAEN	R/W	0h	Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse 0: Disable EPWMxSOCA. 1: Enable EPWMxSOCA pulse. Reset type: SYSRSn

**Table 13-82. 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> <p>Reset type: SYSRSn</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> <p>Reset type: SYSRSn</p>
5	SOCBSELCMP	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 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> <p>Reset type: SYSRSn</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> <p>Reset type: SYSRSn</p>

**Table 13-82. 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 Reset type: SYSRSn
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. Reset type: SYSRSn

### 13.17.2.60 ETPS Register (Offset = A6h) [Reset = 0000h]

ETPS is shown in [Figure 13-152](#) and described in [Table 13-83](#).

Return to the [Summary Table](#).

Event Trigger Pre-Scale Register

**Figure 13-152. ETPS Register**

15	14	13	12	11	10	9	8
SOCBCNT		SOCBPRD		SOCACNT		SOCAPRD	
R-0h		R/W-0h		R-0h		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		SOCPSSEL	INTPSSEL	INTCNT		INTPRD	
R-0-0h		R/W-0h	R/W-0h	R-0h		R/W-0h	

**Table 13-83. ETPS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	SOCBCNT	R	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register These bits indicate how many selected ETSEL[SOCBSEL] events have occurred: 00: No events have occurred. 01: 1 event has occurred. 10: 2 events have occurred. 11: 3 events have occurred. Reset type: SYSRSn
13-12	SOCBPRD	R/W	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits will automatically be cleared. 00: Disable the SOCB event counter. No EPWMxSOCB pulse will be generated 01: Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1 10: Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0 11: Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1 Reset type: SYSRSn
11-10	SOCACNT	R	0h	ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register These bits indicate how many selected ETSEL[SOCASEL] events have occurred: 00: No events have occurred. 01: 1 event has occurred. 10: 2 events have occurred. 11: 3 events have occurred. Reset type: SYSRSn

**Table 13-83. ETPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	SOCAPRD	R/W	0h	<p>ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select</p> <p>These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits will automatically be cleared.</p> <p>00: Disable the SOCA event counter. No EPWMxSOCA pulse will be generated</p> <p>01: Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1</p> <p>10: Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0</p> <p>11: Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1</p> <p>Reset type: SYSRSn</p>
7-6	RESERVED	R-0	0h	Reserved
5	SOCPSSEL	R/W	0h	<p>EPWMxSOC A/B Pre-Scale Selection Bits</p> <p>0: Selects ETPS [SOCACNT/SOCBCNT] and [SOCAPRD/SOCBPRD] registers to determine frequency of events (SOC pulse once every 0-3 events).</p> <p>1: Selects ETSOCPS [SOCACNT2/SOCBCNT2] and [SOCAPRD2/SOCBPRD2] registers to determine frequency of events (SOC pulse once every 0-15 events).</p> <p>Reset type: SYSRSn</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> <p>Reset type: SYSRSn</p>
3-2	INTCNT	R	0h	<p>ePWM Interrupt Event (EPWMx_INT) Counter Register</p> <p>These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].</p> <p>00: No events have occurred.</p> <p>01: 1 event has occurred.</p> <p>10: 2 events have occurred.</p> <p>11: 3 events have occurred.</p> <p>Reset type: SYSRSn</p>

**Table 13-83. ETPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1-0	INTPRD	R/W	0h	<p>ePWM Interrupt (EPWMx_INT) Period Select</p> <p>These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared.</p> <p>Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear.</p> <p>Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.</p> <p>00: Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.</p> <p>01: Generate an interrupt on the first event INTCNT = 01 (first event)</p> <p>10: Generate interrupt on ETPS[INTCNT] = 1,0 (second event)</p> <p>11: Generate interrupt on ETPS[INTCNT] = 1,1 (third event)</p> <p>Reset type: SYSRSn</p>

### 13.17.2.61 ETFLG Register (Offset = A8h) [Reset = 0000h]

ETFLG is shown in [Figure 13-153](#) and described in [Table 13-84](#).

Return to the [Summary Table](#).

Event Trigger Flag Register

**Figure 13-153. ETFLG Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0h	R-0h	R-0-0h	R-0h

**Table 13-84. ETFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R	0h	Latched ePWM ADC Start-of-Conversion A (EPWMxSOCB) Status Flag Unlike the ETFLG[INT] flag, the EPWMxSOCB output will continue to pulse even if the flag bit is set. 0: Indicates no event occurred 1: Indicates that a start of conversion pulse was generated on EPWMxSOCB. The EPWMxSOCB output will continue to be generated even if the flag bit is set. Reset type: SYSRSn
2	SOCA	R	0h	Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag Unlike the ETFLG[INT] flag, the EPWMxSOCA output will continue to pulse even if the flag bit is set. 0: Indicates no event occurred 1: Indicates that a start of conversion pulse was generated on EPWMxSOCA. The EPWMxSOCA output will continue to be generated even if the flag bit is set. Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R	0h	Latched ePWM Interrupt (EPWMx_INT) Status Flag 0: Indicates no event occurred 1: Indicates that an ePWMx interrupt (EPWMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Reset type: SYSRSn

### 13.17.2.62 ETCLR Register (Offset = AAh) [Reset = 0000h]

ETCLR is shown in [Figure 13-154](#) and described in [Table 13-85](#).

Return to the [Summary Table](#).

Event Trigger Clear Register

**Figure 13-154. ETCLR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0/W1S-0h	R-0/W1S-0h	R-0-0h	R-0/W1S-0h

**Table 13-85. ETCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCB) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCB] flag bit Reset type: SYSRSn
2	SOCA	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCA] flag bit Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R-0/W1S	0h	ePWM Interrupt (EPWMx_INT) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated Reset type: SYSRSn

### 13.17.2.63 ETFRC Register (Offset = ACh) [Reset = 0000h]

ETFRC is shown in [Figure 13-155](#) and described in [Table 13-86](#).

Return to the [Summary Table](#).

Event Trigger Force Register

**Figure 13-155. ETFRC Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0-0h				R-0/W1S-0h	R-0/W1S-0h	R-0-0h	R-0/W1S-0h

**Table 13-86. ETFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R-0/W1S	0h	SOCB Force Bit 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. 0: Writing 0 to this bit will be ignored. Always reads back a 0. 1: Generates a pulse on EPWMxSOCB and set the SOCBFLG bit. This bit is used for test purposes. Reset type: SYSRSn
2	SOCA	R-0/W1S	0h	SOCA Force Bit 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. 0: Writing 0 to this bit will be ignored. Always reads back a 0. 1: Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes. Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R-0/W1S	0h	INT Force Bit The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless. 0: Writing 0 to this bit will be ignored. Always reads back a 0. 1: Generates an interrupt on EPWMxINT and set the INT flag bit. This bit is used for test purposes. Reset type: SYSRSn

### 13.17.2.64 ETINTPS Register (Offset = AEh) [Reset = 0000h]

ETINTPS is shown in [Figure 13-156](#) and described in [Table 13-87](#).

Return to the [Summary Table](#).

Event-Trigger Interrupt Pre-Scale Register

**Figure 13-156. ETINTPS Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0-0h							
7	6	5	4	3	2	1	0
INTCNT2				INTPRD2			
R-0h				R/W-0h			

**Table 13-87. 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 Reset type: SYSRSn
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) Reset type: SYSRSn

### 13.17.2.65 ETSOCPS Register (Offset = B0h) [Reset = 0000h]

ETSOCPS is shown in [Figure 13-157](#) and described in [Table 13-88](#).

Return to the [Summary Table](#).

Event-Trigger SOC Pre-Scale Register

**Figure 13-157. ETSOCPS Register**

15	14	13	12	11	10	9	8
SOCBCNT2				SOCBPRD2			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
SOCACNT2				SOCAPRD2			
R-0h				R/W-0h			

**Table 13-88. 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 Reset type: SYSRSn
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 SOC pulse on SOCBCNT2 = 1 (first event) 0010: Generate SOC pulse on SOCBCNT2 = 2 (second event) 0011: Generate SOC pulse on SOCBCNT2 = 3 (third event) 0100: Generate SOC pulse on SOCBCNT2 = 4 (fourth event) ... 1111: Generate SOC pulse on SOCBCNT2 = 15 (fifteenth event) Reset type: SYSRSn
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 Reset type: SYSRSn

**Table 13-88. ETSOCPS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	SOCAPRD2	R/W	0h	EPWMxSOCA Period 2 Select When ETPS[SOCPSSEL] = 1, these bits select how many selected event need to occur before an SOCA pulse is generated: 0000: Disable counter 0001: Generate SOC pulse on SOCACNT2 = 1 (first event) 0010: Generate SOC pulse on SOCACNT2 = 2 (second event) 0011: Generate SOC pulse on SOCACNT2 = 3 (third event) 0100: Generate SOC pulse on SOCACNT2 = 4 (fourth event) ... 1111: Generate SOC pulse on SOCACNT2 = 15 (fifteenth event) Reset type: SYSRSn

### 13.17.2.66 ETCNTINITCTL Register (Offset = B2h) [Reset = 0000h]

ETCNTINITCTL is shown in [Figure 13-158](#) and described in [Table 13-89](#).

Return to the [Summary Table](#).

Event-Trigger Counter Initialization Control Register

**Figure 13-158. 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-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0-0h			
7		6		5		4		3		2		1		0	
RESERVED															
R-0-0h															

**Table 13-89. ETCNTINITCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SOCBINITEN	R/W	0h	EPWMxSOCB Counter 2 Initialization Enable 0: Has no effect. 1: Enable initialization of EPWMxSOCB counter with contents of ETCNTINIT[SOCBINIT] on a SYNC event or software force. Reset type: SYSRSn
14	SOCAINITEN	R/W	0h	EPWMxSOCA Counter 2 Initialization Enable 0: Has no effect. 1: Enable initialization of EPWMxSOCA counter with contents of ETCNTINIT[SOCAINIT] on a SYNC event or software force. Reset type: SYSRSn
13	INTINITEN	R/W	0h	EPWMxINT Counter 2 Initialization Enable 0: Has no effect. 1: Enable initialization of EPWMxINT counter 2 with contents of ETCNTINIT[INTINIT] on a SYNC event or software force. Reset type: SYSRSn
12	SOCBINITFRC	R-0/W1S	0h	EPWMxSOCB Counter 2 Initialization Force 0: Has no effect. 1: This bit forces the ET EPWMxSOCB counter to be initialized with the contents of ETCNTINIT[SOCBINIT]. Reset type: SYSRSn
11	SOCAINITFRC	R-0/W1S	0h	EPWMxSOCA Counter 2 Initialization Force 0: Has no effect. 1: This bit forces the ET EPWMxSOCA counter to be initialized with the contents of ETCNTINIT[SOCAINIT]. Reset type: SYSRSn
10	INTINITFRC	R-0/W1S	0h	EPWMxINT Counter 2 Initialization Force 0: Has no effect. 1: This bit forces the ET EPWMxINT counter to be initialized with the contents of ETCNTINIT[INTINIT]. Reset type: SYSRSn
9-0	RESERVED	R-0	0h	Reserved

### 13.17.2.67 ETCNTINIT Register (Offset = B4h) [Reset = 0000h]

ETCNTINIT is shown in [Figure 13-159](#) and described in [Table 13-90](#).

Return to the [Summary Table](#).

Event-Trigger Counter Initialization Register

**Figure 13-159. ETCNTINIT Register**

15	14	13	12	11	10	9	8
RESERVED				SOCBINIT			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
SOCAINIT				INTINIT			
R/W-0h				R/W-0h			

**Table 13-90. ETCNTINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	SOCBINIT	R/W	0h	EPWMxSOCB Counter 2 Initialization Bits The ET EPWMxSOCB counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn
7-4	SOCAINIT	R/W	0h	EPWMxSOCA Counter 2 Initialization Bits The ET EPWMxSOCA counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn
3-0	INTINIT	R/W	0h	EPWMxINT Counter 2 Initialization Bits The ET EPWMxINT counter is initialized with the contents of this register on an ePWM SYNC event or a software force. Reset type: SYSRSn

### 13.17.2.68 DCTRIPSEL Register (Offset = C0h) [Reset = 0000h]

DCTRIPSEL is shown in [Figure 13-160](#) and described in [Table 13-91](#).

Return to the [Summary Table](#).

Digital Compare Trip Select Register

**Figure 13-160. DCTRIPSEL Register**

15	14	13	12	11	10	9	8
DCBLCOMPSEL				DCBHCOMPSEL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
DCALCOMPSEL				DCAHCOMPSEL			
R/W-0h				R/W-0h			

**Table 13-91. DCTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	DCBLCOMPSEL	R/W	0h	Digital Compare B Low Input Select Bits 0000: TRIPIN1 0001: TRIPIN2 0010: TRIPIN3 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCBLTRIPSEL register ORed together) Reset type: SYSRSn
11-8	DCBHCOMPSEL	R/W	0h	Digital Compare B High Input Select Bits 0000: TRIPIN1 0001: TRIPIN2 0010: TRIPIN3 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCBHTRIPSEL register ORed together) Reset type: SYSRSn
7-4	DCALCOMPSEL	R/W	0h	Digital Compare A Low Input Select Bits 0000: TRIPIN1 0001: TRIPIN2 0010: TRIPIN3 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCALTRIPSEL register ORed together) Reset type: SYSRSn

**Table 13-91. 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 0001: TRIPIN2 0010: TRIPIN3 0011: TRIPIN4 ... 1011: TRIPIN12 1100: Reserved 1101: TRIPIN14 1110: TRIPIN15 1111: Trip combination input (all trip inputs selected by DCAHTRIPSEL register ORed together) Reset type: SYSRSn

### 13.17.2.69 DCACTL Register (Offset = C3h) [Reset = 0000h]

DCACTL is shown in [Figure 13-161](#) and described in [Table 13-92](#).

Return to the [Summary Table](#).

Digital Compare A Control Register

**Figure 13-161. DCACTL Register**

15	14	13	12	11	10	9	8
EVT2LAT	EVT2LATCLRSEL		EVT2LATSEL	RESERVED		EVT2FRCSYN CSEL	EVT2SRCSEL
R-0h	R/W-0h		R/W-0h	R-0-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
EVT1LAT	EVT1LATCLRSEL		EVT1LATSEL	EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL
R-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-92. DCACTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	EVT2LAT	R	0h	Indicates the status of DCAEVT2LAT signal. 0 : The DCAEVT2LAT latch is cleared. 1 : The DCAEVT2LAT latch is set. Reset type: SYSRSn
14-13	EVT2LATCLRSEL	R/W	0h	DCAEVT2 Latched clear source select: 00: CNT_ZERO event clears DCAEVT2 latch. 01: PRD_EQ event clears DCAEVT2 latch. 10: CNT_ZERO event or PRD_EQ event clears DCAEVT2 latch. 11: Reserved. Reset type: SYSRSn
12	EVT2LATSEL	R/W	0h	DCAEVT2 Latched signal select: 0: Does not select the DCAEVT2 latched signal as source of DCAEVT2.force. 1: Selects the DCAEVT2 latched signal as source of DCAEVT2.force. Reset type: SYSRSn
11-10	RESERVED	R-0	0h	Reserved
9	EVT2FRCSYNSEL	R/W	0h	DCAEVT2 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn
8	EVT2SRCSEL	R/W	0h	DCAEVT2 Source Signal Select 0: Source Is DCAEVT2 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn
7	EVT1LAT	R	0h	Indicates the status of DCAEVT1LAT signal. 0 : The DCAEVT1LAT latch is cleared. 1 : The DCAEVT1LAT latch is set. Reset type: SYSRSn
6-5	EVT1LATCLRSEL	R/W	0h	DCAEVT1 Latched clear source select: 00: CNT_ZERO event clears DCAEVT1 latch. 01: PRD_EQ event clears DCAEVT1 latch. 10: CNT_ZERO event or PRD_EQ event clears DCAEVT1 latch. 11 : Reserved. Reset type: SYSRSn

**Table 13-92. DCACTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EVT1LATSEL	R/W	0h	DCAEVT1 Latched signal select: 0: Does not select the DCAEVT1 latched signal as source of DCAEVT1.force. 1: Selects the DCAEVT1 latched signal as source of DCAEVT1.force. Reset type: SYSRSn
3	EVT1SYNCE	R/W	0h	DCAEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled Reset type: SYSRSn
2	EVT1SOCE	R/W	0h	DCAEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled Reset type: SYSRSn
1	EVT1FRCSYNCSSEL	R/W	0h	DCAEVT1 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn
0	EVT1SRCSEL	R/W	0h	DCAEVT1 Source Signal Select 0: Source Is DCAEVT1 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn

### 13.17.2.70 DCBCTL Register (Offset = C4h) [Reset = 0000h]

DCBCTL is shown in [Figure 13-162](#) and described in [Table 13-93](#).

Return to the [Summary Table](#).

Digital Compare B Control Register

**Figure 13-162. DCBCTL Register**

15	14	13	12	11	10	9	8
EVT2LAT	EVT2LATCLRSEL	EVT2LATSEL	RESERVED		EVT2FRCSYN CSEL	EVT2SRCSEL	
R-0h	R/W-0h	R/W-0h	R-0-0h		R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
EVT1LAT	EVT1LATCLRSEL	EVT1LATSEL	EVT1SYNCE	EVT1SOCE	EVT1FRCSYN CSEL	EVT1SRCSEL	
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-93. DCBCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	EVT2LAT	R	0h	Indicates the status of DCBEVT2LAT signal. 0 The DCBEVT2LAT latch is cleared. 1 The DCBEVT2LAT latch is set. Reset type: SYSRSn
14-13	EVT2LATCLRSEL	R/W	0h	DCBEVT2 Latched clear source select: 00 CNT_ZERO event clears DCBEVT2 latch. 01 PRD_EQ event clears DCBEVT2 latch. 10 CNT_ZERO event or PRD_EQ event clears DCBEVT2 latch. 11 Reserved. Reset type: SYSRSn
12	EVT2LATSEL	R/W	0h	DCBEVT2 Latched signal select: 0 Does not select the DCBEVT2 latched signal (Refer figure 'Modifications to DCBEVT1.force/DCBEVT2.force generation.') as source of DCBEVT2.force. 1 Selects the DCBEVT2 latched signal as source of DCBEVT2.force. Reset type: SYSRSn
11-10	RESERVED	R-0	0h	Reserved
9	EVT2FRCSYNSEL	R/W	0h	DCBEVT2 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn
8	EVT2SRCSEL	R/W	0h	DCBEVT2 Source Signal Select 0: Source Is DCBEVT2 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn
7	EVT1LAT	R	0h	Indicates the status of DCBEVT1LAT signal. 0 The DCBEVT1LAT latch is cleared. 1 The DCBEVT1LAT latch is set. Reset type: SYSRSn
6-5	EVT1LATCLRSEL	R/W	0h	DCBEVT1 Latched clear source select: 00 CNT_ZERO event clears DCBEVT1 latch. 01 PRD_EQ event clears DCBEVT1 latch. 10 CNT_ZERO event or PRD_EQ event clears DCBEVT1 latch. 11 Reserved. Reset type: SYSRSn

**Table 13-93. DCBCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	EVT1LATSEL	R/W	0h	DCBEVT1 Latched signal select: 0 Does not select the DCBEVT1 latched signal (Refer figure 'Modifications to DCBEVT1.force/DCBEVT2.force generation.') as source of DCBEVT1.force. 1 Selects the DCBEVT1 latched signal as source of DCBEVT1.force. Reset type: SYSRSn
3	EVT1SYNCE	R/W	0h	DCBEVT1 SYNC, Enable/Disable 0: SYNC Generation Disabled 1: SYNC Generation Enabled Reset type: SYSRSn
2	EVT1SOCE	R/W	0h	DCBEVT1 SOC, Enable/Disable 0: SOC Generation Disabled 1: SOC Generation Enabled Reset type: SYSRSn
1	EVT1FRCSYNCSSEL	R/W	0h	DCBEVT1 Force Synchronization Signal Select 0: Source is synchronized with EPWMCLK 1: Source is passed through asynchronously Reset type: SYSRSn
0	EVT1SRCSEL	R/W	0h	DCBEVT1 Source Signal Select 0: Source Is DCBEVT1 Signal 1: Source Is DCEVTFILT Signal Reset type: SYSRSn

### 13.17.2.71 DCFCTL Register (Offset = C7h) [Reset = 0000h]

DCFCTL is shown in [Figure 13-163](#) and described in [Table 13-94](#).

Return to the [Summary Table](#).

Digital Compare Filter Control Register

**Figure 13-163. DCFCTL Register**

15	14	13	12	11	10	9	8
EDGESTATUS			EDGECOUNT			EDGEMODE	
R-0h			R/W-0h			R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	EDGEFILTSEL	PULSESEL		BLANKINV	BLANKE	SRCSEL	
R-0-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	

**Table 13-94. DCFCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	EDGESTATUS	R	0h	Edge Status: These bits reflect the total number of edges currently captured. When the value matches the EDGECOUNT, the status bits are set to zero, and a TBCLK wide pulse is generated which can then be output on the DCEVTFILT signal. The edge counter can be reset by writing 000 to the EDGECOUNT value. Reset type: SYSRSn
12-10	EDGECOUNT	R/W	0h	Edge Count: These bits select how many edges to count before generating a TBCLK wide pulse on the DCEVTFILT signal: 000: no edges, reset current EDGESTATUS bits to 0,0,0 001: 1 edge 010: 2 edges 011: 3 edges 100: 4 edges 101: 5 edges 110: 6 edges 111: 7 edges Reset type: SYSRSn
9-8	EDGEMODE	R/W	0h	Edge Mode Select: 00: Low To High Edge 01: High To Low Edge 10: Both Edges 11: Reserved Reset type: SYSRSn
7	RESERVED	R-0	0h	Reserved
6	EDGEFILTSEL	R/W	0h	Edge Filter Select: 0: Edge Filter Not Selected 1: Edge Filter Selected Reset type: SYSRSn
5-4	PULSESEL	R/W	0h	Pulse Select For Blanking & Capture Alignment 00: Time-base counter equal to period (TBCTR = TBPRD) 01: Time-base counter equal to zero (TBCTR = 0x00) 10: Time-base counter equal to zero (TBCTR = 0x00) or period (TBCTR = TBPRD) 11: BLANKPULSEMIX Reset type: SYSRSn
3	BLANKINV	R/W	0h	Blanking Window Inversion 0: Blanking window not inverted 1: Blanking window inverted Reset type: SYSRSn

**Table 13-94. DCFCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	BLANKE	R/W	0h	Blanking Window Enable/Disable 0: Blanking window is disabled 1: Blanking window is enabled Reset type: SYSRSn
1-0	SRCSEL	R/W	0h	Filter Block Signal Source Select 00: Source Is DCAEVT1 Signal 01: Source Is DCAEVT2 Signal 10: Source Is DCBEVT1 Signal 11: Source Is DCBEVT2 Signal Reset type: SYSRSn

### 13.17.2.72 DCCAPCTL Register (Offset = C8h) [Reset = 0000h]

DCCAPCTL is shown in [Figure 13-164](#) and described in [Table 13-95](#).

Return to the [Summary Table](#).

Digital Compare Capture Control Register

**Figure 13-164. DCCAPCTL Register**

15		14		13		12		11		10		9		8	
CAPMODE		CAPCLR		CAPSTS		RESERVED									
R/W-0h		R-0/W1S-0h		R-0h		R-0-0h									
7		6		5		4		3		2		1		0	
RESERVED												SHDWMODE		CAPE	
R-0-0h												R/W-0h		R/W-0h	

**Table 13-95. DCCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CAPMODE	R/W	0h	<p>Counter Capture Mode</p> <p>0: When a DCEVTFILT occurs and the counter capture is enabled, then the current TBCNT value is captured in the active register. When the respective trip event occurs, further trip (capture) events are ignored until the next PRD_eq or CNT_zero event (as selected by the PULSESEL bit in the DCFCTL register) re-triggers the capture mechanism.</p> <p>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.</p> <p>If shadow mode is enabled, via SHDWMODE bit in DCCAPCTL register, the active register is copied to the shadow register on the PRD_eq or CNT_zero event (whichever is selected by PULSESEL bit in DCFCTL register). CPU reads of this register will return the shadow register value.</p> <p>1: When a DCEVTFILT occurs and the counter capture is enabled, then the current TBCNT value is captured in the active register. When the respective trip event occurs - it will set the CAPSTS flag and further trip (capture) events are ignored until this bit is cleared. CAPSTS can be cleared by writing to CAPCLR bit in DCCAPCTL register and it re-triggers the capture mechanism.</p> <p>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.</p> <p>If shadow mode is enabled, via SHDWMODE bit in DCCAPCTL register, the active register is copied to the shadow register on the PRD_eq or CNT_zero event (whichever is selected by PULSESEL bit in DCFCTL register). CPU reads of this register will return the shadow register value.</p> <p>Reset type: SYSRSn</p>
14	CAPCLR	R-0/W1S	0h	<p>DC Capture Latched Status Clear Flag</p> <p>0: Writing a 0 has no effect.</p> <p>1: Writing a 1 will clear this CAPSTS (set) condition.</p> <p>Reset type: SYSRSn</p>
13	CAPSTS	R	0h	<p>Latched Status Flag for Capture Event</p> <p>0: No DC capture event occurred.</p> <p>1: A DC capture event has occurred.</p> <p>Reset type: SYSRSn</p>
12-2	RESERVED	R-0	0h	Reserved

**Table 13-95. DCCAPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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. Reset type: SYSRSn
0	CAPE	R/W	0h	TBCTR Counter Capture Enable/Disable 0: Disable the time-base counter capture. 1: Enable the time-base counter capture. Reset type: SYSRSn

### 13.17.2.73 DCFOFFSET Register (Offset = C9h) [Reset = 0000h]

DCFOFFSET is shown in [Figure 13-165](#) and described in [Table 13-96](#).

Return to the [Summary Table](#).

Digital Compare Filter Offset Register

**Figure 13-165. DCFOFFSET Register**

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

**Table 13-96. DCFOFFSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFOFFSET	R/W	0h	Blanking Window Offset 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. Reset type: SYSRSn

### 13.17.2.74 DCFOFFSETCNT Register (Offset = CAh) [Reset = 0000h]

DCFOFFSETCNT is shown in [Figure 13-166](#) and described in [Table 13-97](#).

Return to the [Summary Table](#).

Digital Compare Filter Offset Counter Register

**Figure 13-166. DCFOFFSETCNT Register**

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

**Table 13-97. DCFOFFSETCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFOFFSETCNT	R	0h	<p>Blanking Offset Counter</p> <p>These 16-bits are read only and indicate the current value of the offset counter. The counter counts down to zero and then stops until it is re-loaded on the next period or zero event as defined by the DCCTL[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 an emulation stop.</p> <p>Reset type: SYSRSn</p>

### 13.17.2.75 DCFWINDOW Register (Offset = CBh) [Reset = 0000h]

DCFWINDOW is shown in [Figure 13-167](#) and described in [Table 13-98](#).

Return to the [Summary Table](#).

Digital Compare Filter Window Register

**Figure 13-167. DCFWINDOW Register**

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

**Table 13-98. DCFWINDOW Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFWINDOW	R/W	0h	Blanking Window Width 00h: No blanking window is generated. 01-FFFFh: Specifies the width of the blanking window in TBCLK cycles. The blanking window begins when the offset counter expires. When this occurs, the window counter is loaded and begins to count down. If the blanking window is currently active and the offset counter expires, the blanking window counter is not restarted and the blanking window is cut short prematurely. Care should be taken to avoid this situation. The blanking window can cross a PWM period boundary. Reset type: SYSRSn

### 13.17.2.76 DCFWINDOWCNT Register (Offset = CCh) [Reset = 0000h]

DCFWINDOWCNT is shown in [Figure 13-168](#) and described in [Table 13-99](#).

Return to the [Summary Table](#).

Digital Compare Filter Window Counter Register

**Figure 13-168. DCFWINDOWCNT Register**

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

**Table 13-99. DCFWINDOWCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCFWINDOWCNT	R	0h	Blanking Window Counter These 16 bits are read only and indicate the current value of the window counter. The counter counts down to zero and then stops until it is re-loaded when the offset counter reaches zero again. Reset type: SYSRSn

**13.17.2.77 BLANKPULSEMIXSEL Register (Offset = CDh) [Reset = 0000h]**

 BLANKPULSEMIXSEL is shown in [Figure 13-169](#) and described in [Table 13-100](#).

 Return to the [Summary Table](#).

Blanking window trigger pulse select register

**Figure 13-169. BLANKPULSEMIXSEL Register**

15	14	13	12	11	10	9	8
RESERVED						CDD	CDU
R-0-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CCD	CCU	CBD	CBU	CAD	CAU	PRD	ZRO
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-100. BLANKPULSEMIXSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R-0	0h	Reserved
9	CDD	R/W	0h	Enable event time-base counter equal to CMPD when the timer is decrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPD down-count match enable event is not enabled 1: Enable CMPD down-count match enable event Reset type: SYSRSn
8	CDU	R/W	0h	Enable event time-base counter equal to CMPD when the timer is incrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPD up-count match enable event is not enabled 1: Enable CMPD up-count match enable event Reset type: SYSRSn
7	CCD	R/W	0h	Enable event time-base counter equal to CMPC when the timer is decrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPC down-count match enable event is not enabled 1: Enable CMPC down-count match enable event Reset type: SYSRSn
6	CCU	R/W	0h	Enable event time-base counter equal to CMPC when the timer is incrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPC up-count match enable event is not enabled 1: Enable CMPC up-count match enable event Reset type: SYSRSn
5	CBD	R/W	0h	Enable event time-base counter equal to CMPB when the timer is decrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPB down-count match enable event is not enabled 1: Enable CMPB down-count match enable event Reset type: SYSRSn
4	CBU	R/W	0h	Enable event time-base counter equal to CMPB when the timer is incrementing to the mixed ET interrupt trigger signal (BLANKPULSEMIX). 0: CMPB up-count match enable event is not enabled 1: Enable CMPB up-count match enable event Reset type: SYSRSn
3	CAD	R/W	0h	Enable event time-base counter equal to CMPA when the timer is decrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPA down-count match enable event is not enabled 1: Enable CMPA down-count match enable event Reset type: SYSRSn

**Table 13-100. BLANKPULSEMIXSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	CAU	R/W	0h	Enable event time-base counter equal to CMPA when the timer is incrementing to the blanking window trigger (BLANKPULSEMIX). 0: CMPA up-count match enable event is not enabled 1: Enable CMPA up-count match enable event Reset type: SYSRSn
1	PRD	R/W	0h	Enable event time-base counter equal to period (TBCTR = TBPRD) to the blanking window trigger (BLANKPULSEMIX). 0: Period match event is not enabled 1: Enable period match event Reset type: SYSRSn
0	ZRO	R/W	0h	Enable event time-base counter equal to zero (TBCTR = 0x00) to the blanking window trigger (BLANKPULSEMIX). 0: Zero match event is not enabled 1: Enable zero match event Reset type: SYSRSn

### 13.17.2.78 DCCAP Register (Offset = CFh) [Reset = 0000h]

DCCAP is shown in [Figure 13-170](#) and described in [Table 13-101](#).

Return to the [Summary Table](#).

Digital Compare Counter Capture Register

**Figure 13-170. DCCAP Register**

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

**Table 13-101. DCCAP Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	DCCAP	R	0h	Digital Compare Time-Base Counter Capture 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. - 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. Reset type: SYSRSn

### 13.17.2.79 DCAHTRIPSEL Register (Offset = D2h) [Reset = 0000h]

DCAHTRIPSEL is shown in [Figure 13-171](#) and described in [Table 13-102](#).

Return to the [Summary Table](#).

Digital Compare AH Trip Select

**Figure 13-171. DCAHTRIPSEL Register**

15	14	13	12	11	10	9	8
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-102. DCAHTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAH mux Reset type: SYSRSn

**Table 13-102. DCAHTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAH mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAH mux Reset type: SYSRSn

### 13.17.2.80 DCALTRIPSEL Register (Offset = D3h) [Reset = 0000h]

DCALTRIPSEL is shown in [Figure 13-172](#) and described in [Table 13-103](#).

Return to the [Summary Table](#).

Digital Compare AL Trip Select

**Figure 13-172. DCALTRIPSEL Register**

15	14	13	12	11	10	9	8
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-103. DCALTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**Table 13-103. DCALTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

### 13.17.2.81 DCBHTRIPSEL Register (Offset = D4h) [Reset = 0000h]

DCBHTRIPSEL is shown in [Figure 13-173](#) and described in [Table 13-104](#).

Return to the [Summary Table](#).

Digital Compare BH Trip Select

**Figure 13-173. DCBHTRIPSEL Register**

15	14	13	12	11	10	9	8
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-104. DCBHTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCBH mux Reset type: SYSRSn

**Table 13-104. DCBHTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCBH mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCBH mux Reset type: SYSRSn

### 13.17.2.82 DCBLTRIPSEL Register (Offset = D5h) [Reset = 0000h]

DCBLTRIPSEL is shown in [Figure 13-174](#) and described in [Table 13-105](#).

Return to the [Summary Table](#).

Digital Compare BL Trip Select

**Figure 13-174. DCBLTRIPSEL Register**

15	14	13	12	11	10	9	8
RESERVED	TRIPINPUT15	TRIPINPUT14	RESERVED	TRIPINPUT12	TRIPINPUT11	TRIPINPUT10	TRIPINPUT9
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TRIPINPUT8	TRIPINPUT7	TRIPINPUT6	TRIPINPUT5	TRIPINPUT4	TRIPINPUT3	TRIPINPUT2	TRIPINPUT1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 13-105. DCBLTRIPSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	TRIPINPUT15	R/W	0h	TRIP Input 15 0: Trip Input 15 not selected as combinational ORed input 1: Trip Input 15 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
13	TRIPINPUT14	R/W	0h	TRIP Input 14 0: Trip Input 14 not selected as combinational ORed input 1: Trip Input 14 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
12	RESERVED	R/W	0h	Reserved
11	TRIPINPUT12	R/W	0h	TRIP Input 12 0: Trip Input 12 not selected as combinational ORed input 1: Trip Input 12 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
10	TRIPINPUT11	R/W	0h	TRIP Input 11 0: Trip Input 11 not selected as combinational ORed input 1: Trip Input 11 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
9	TRIPINPUT10	R/W	0h	TRIP Input 10 0: Trip Input 10 not selected as combinational ORed input 1: Trip Input 10 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
8	TRIPINPUT9	R/W	0h	TRIP Input 9 0: Trip Input 9 not selected as combinational ORed input 1: Trip Input 9 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
7	TRIPINPUT8	R/W	0h	TRIP Input 8 0: Trip Input 8 not selected as combinational ORed input 1: Trip Input 8 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
6	TRIPINPUT7	R/W	0h	TRIP Input 7 0: Trip Input 7 not selected as combinational ORed input 1: Trip Input 7 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
5	TRIPINPUT6	R/W	0h	TRIP Input 6 0: Trip Input 6 not selected as combinational ORed input 1: Trip Input 6 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

**Table 13-105. DCBLTRIPSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	TRIPINPUT5	R/W	0h	TRIP Input 5 0: Trip Input 5 not selected as combinational ORed input 1: Trip Input 5 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
3	TRIPINPUT4	R/W	0h	TRIP Input 4 0: Trip Input 4 not selected as combinational ORed input 1: Trip Input 4 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
2	TRIPINPUT3	R/W	0h	TRIP Input 3 0: Trip Input 3 not selected as combinational ORed input 1: Trip Input 3 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
1	TRIPINPUT2	R/W	0h	TRIP Input 2 0: Trip Input 2 not selected as combinational ORed input 1: Trip Input 2 selected as combinational ORed input to DCAL mux Reset type: SYSRSn
0	TRIPINPUT1	R/W	0h	TRIP Input 1 0: Trip Input 1 not selected as combinational ORed input 1: Trip Input 1 selected as combinational ORed input to DCAL mux Reset type: SYSRSn

### 13.17.2.83 EPWMLOCK Register (Offset = FAh) [Reset = 0000000h]

EPWMLOCK is shown in [Figure 13-175](#) and described in [Table 13-106](#).

Return to the [Summary Table](#).

EPWM Lock Register

**Figure 13-175. EPWMLOCK Register**

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

**Table 13-106. EPWMLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	KEY	R-0/W	0h	Write to this register succeeds only if this field is written with a value of 0xa5a5 Note: [1] Due to this KEY, only 32-bit writes will succeed (provided the KEY matches). 16-bit writes to the upper or lower half of this register will be ignored Reset type: SYSRSn
15-5	RESERVED	R	0h	Reserved
4	DCLOCK	R/WOnce	0h	0: Digital Compare registers from 0xC0 to 0xD9 offsets are protected by EALLOW. 1: Digital Compare registers from 0xC0 to 0xD9 offsets are locked and not writable. Reset type: SYSRSn
3	TZCLRLOCK	R/WOnce	0h	0: Trip Zone registers from 0x97 to 0x9B offsets are protected by EALLOW. 1: Trip Zone registers from 0x97 to 0x9B offsets are locked and not writable. Reset type: SYSRSn
2	TZCFGLOCK	R/WOnce	0h	0: TripZone registers from 0x80 to 0x8D and TZTRIPOUTSEL at 0x9D offsets are protected by EALLOW. 1: TripZone registers from 0x80 to 0x8D and TZTRIPOUTSEL at 0x9D offsets are locked and not writable. Reset type: SYSRSn
1	GLLOCK	R/WOnce	0h	0: Global Load registers from 0x34 to 0x35 offsets are protected by EALLOW. 1: Global Load registers from 0x34 to 0x35 offsets are locked and not writable Reset type: SYSRSn

**Table 13-106. EPWMLOCK Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	HRLOCK	R/WOnce	0h	0: HRPWM registers from 0x20 to 0x2D offsets are protected by EALLOW 1: HRPWM registers from 0x20 and 0x2D offsets are locked and not writable. Reset type: SYSRSn

### 13.17.2.84 HWVDELVAL Register (Offset = FDh) [Reset = 0000h]

HWVDELVAL is shown in [Figure 13-176](#) and described in [Table 13-107](#).

Return to the [Summary Table](#).

Hardware Valley Mode Delay Register

**Figure 13-176. HWVDELVAL Register**

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

**Table 13-107. HWVDELVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	HWVDELVAL	R	0h	Hardware Valley Delay Value Register This read only register reflects the hardware delay value calculated by the equations defined in VCAPCTL[VDELAYDIV]. This reflects the latest value from the hardware calculations and can change every time valley capture sequence is triggered and VCAP1 and VCAP2 values are updated. Reset type: SYSRSn

### 13.17.2.85 VCNTVAL Register (Offset = FEh) [Reset = 0000h]

VCNTVAL is shown in [Figure 13-177](#) and described in [Table 13-108](#).

Return to the [Summary Table](#).

Hardware Valley Counter Register

**Figure 13-177. VCNTVAL Register**

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

**Table 13-108. VCNTVAL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	VCNTVAL	R	0h	Valley Time Base Counter Register This register reflects the captured VCNT value upon occurrence of STOPEGE selected in VCNTCFG register. Reset type: SYSRSn

### 13.17.3 Register to Driverlib Function Mapping

#### 13.17.3.1 EPWM Registers to Driverlib Functions

**Table 13-109. EPWM Registers to Driverlib Functions**

File	Driverlib Function
<b>TBCTL</b>	
epwm.c	EPWM_setEmulationMode
epwm.h	EPWM_setCountModeAfterSync
epwm.h	EPWM_setClockPrescaler
epwm.h	EPWM_forceSyncPulse
epwm.h	EPWM_setOneShotSyncOutTrigger
epwm.h	EPWM_setPeriodLoadMode
epwm.h	EPWM_enablePhaseShiftLoad
epwm.h	EPWM_disablePhaseShiftLoad
epwm.h	EPWM_setTimeBaseCounterMode
epwm.h	EPWM_selectPeriodLoadEvent
epwm.h	EPWM_enableOneShotSync
epwm.h	EPWM_disableOneShotSync
epwm.h	EPWM_startOneShotSync
<b>TBCTL2</b>	
epwm.h	EPWM_selectPeriodLoadEvent
epwm.h	EPWM_enableOneShotSync
epwm.h	EPWM_disableOneShotSync
epwm.h	EPWM_startOneShotSync
<b>SYNCINSEL</b>	
epwm.h	EPWM_setSyncInPulseSource
<b>TBCTR</b>	
epwm.h	EPWM_setTimeBaseCounter
epwm.h	EPWM_getTimeBaseCounterValue
<b>TBSTS</b>	
epwm.h	EPWM_getTimeBaseCounterOverflowStatus
epwm.h	EPWM_clearTimeBaseCounterOverflowEvent
epwm.h	EPWM_getSyncStatus
epwm.h	EPWM_clearSyncEvent
epwm.h	EPWM_getTimeBaseCounterDirection
<b>SYNCOUTEN</b>	
epwm.h	EPWM_enableSyncOutPulseSource
epwm.h	EPWM_disableSyncOutPulseSource
<b>TBCTL3</b>	
epwm.h	EPWM_setOneShotSyncOutTrigger
<b>CMPCTL</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode
epwm.h	EPWM_getCounterCompareShadowStatus
<b>CMPCTL2</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>DBCTL</b>	
epwm.h	EPWM_setDeadBandOutputSwapMode
epwm.h	EPWM_setDeadBandDelayMode
epwm.h	EPWM_setDeadBandDelayPolarity
epwm.h	EPWM_setRisingEdgeDeadBandDelayInput
epwm.h	EPWM_setFallingEdgeDeadBandDelayInput
epwm.h	EPWM_setDeadBandControlShadowLoadMode
epwm.h	EPWM_disableDeadBandControlShadowLoadMode
epwm.h	EPWM_setRisingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_disableRisingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_setFallingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_disableFallingEdgeDelayCountShadowLoadMode
epwm.h	EPWM_setDeadBandCounterClock
<b>DBCTL2</b>	
epwm.h	EPWM_setDeadBandControlShadowLoadMode
epwm.h	EPWM_disableDeadBandControlShadowLoadMode
<b>AQCTL</b>	
epwm.h	EPWM_setActionQualifierShadowLoadMode
epwm.h	EPWM_disableActionQualifierShadowLoadMode
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setActionQualifierActionComplete
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQTSRCSEL</b>	
epwm.h	EPWM_setActionQualifierT1TriggerSource
epwm.h	EPWM_setActionQualifierT2TriggerSource
<b>PCCTL</b>	
epwm.h	EPWM_enableChopper
epwm.h	EPWM_disableChopper
epwm.h	EPWM_setChopperDutyCycle
epwm.h	EPWM_setChopperFreq
epwm.h	EPWM_setChopperFirstPulseWidth
<b>VCAPCTL</b>	
epwm.h	EPWM_enableValleyCapture
epwm.h	EPWM_disableValleyCapture
epwm.h	EPWM_startValleyCapture
epwm.h	EPWM_setValleyTriggerSource
epwm.h	EPWM_enableValleyHWDelay
epwm.h	EPWM_disableValleyHWDelay
epwm.h	EPWM_setValleyDelayDivider
<b>VCNTCFG</b>	
epwm.h	EPWM_setValleyTriggerEdgeCounts
epwm.h	EPWM_getValleyEdgeStatus
<b>HRCNFG</b>	
-	
<b>HRPWR</b>	

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	
<b>HRMSTEP</b>	
-	
<b>HRCNFG2</b>	
-	
<b>HRPCTL</b>	
-	
<b>TRREM</b>	
-	
<b>GLDCTL</b>	
epwm.h	EPWM_enableGlobalLoad
epwm.h	EPWM_disableGlobalLoad
epwm.h	EPWM_setGlobalLoadTrigger
epwm.h	EPWM_setGlobalLoadEventPrescale
epwm.h	EPWM_getGlobalLoadEventCount
epwm.h	EPWM_disableGlobalLoadOneShotMode
epwm.h	EPWM_enableGlobalLoadOneShotMode
epwm.h	EPWM_setGlobalLoadOneShotLatch
epwm.h	EPWM_forceGlobalLoadOneShotEvent
<b>GLDCFG</b>	
epwm.h	EPWM_enableGlobalLoadRegisters
epwm.h	EPWM_disableGlobalLoadRegisters
<b>XLINK</b>	
epwm.h	EPWM_setupEPWMLinks
<b>AQCTLA</b>	
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setActionQualifierActionComplete
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQCTLA2</b>	
epwm.h	EPWM_setActionQualifierAction
epwm.h	EPWM_setAdditionalActionQualifierActionComplete
<b>AQCTLB</b>	
-	See AQCTLA
<b>AQCTLB2</b>	
-	See AQCTLA2
<b>AQSFRC</b>	
epwm.h	EPWM_setActionQualifierContSWForceShadowMode
epwm.h	EPWM_setActionQualifierSWAction
epwm.h	EPWM_forceActionQualifierSWAction
<b>AQCSFRC</b>	
epwm.h	EPWM_setActionQualifierContSWForceAction
<b>DBREDHR</b>	
-	
<b>DBRED</b>	
epwm.h	EPWM_setRisingEdgeDelayCount

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>DBFEDHR</b>	
-	
<b>DBFED</b>	
epwm.h	EPWM_setFallingEdgeDelayCount
<b>TBPHS</b>	
epwm.h	EPWM_setPhaseShift
<b>TBPRDHR</b>	
-	
<b>TBPRD</b>	
epwm.h	EPWM_setTimeBasePeriod
epwm.h	EPWM_getTimeBasePeriod
<b>CMPA</b>	
epwm.h	EPWM_setCounterCompareValue
epwm.h	EPWM_getCounterCompareValue
<b>CMPB</b>	
-	See CMPA
<b>CMPC</b>	
epwm.h	EPWM_setCounterCompareShadowLoadMode
epwm.h	EPWM_disableCounterCompareShadowLoadMode
epwm.h	EPWM_getCounterCompareShadowStatus
<b>CPMD</b>	
-	See CMPC
<b>GLDCTL2</b>	
epwm.h	EPWM_setGlobalLoadOneShotLatch
epwm.h	EPWM_forceGlobalLoadOneShotEvent
<b>SWVDELVAL</b>	
epwm.h	EPWM_setValleySWDelayValue
<b>TZSEL</b>	
epwm.h	EPWM_enableTripZoneSignals
epwm.h	EPWM_disableTripZoneSignals
<b>TZDCSEL</b>	
epwm.h	EPWM_setTripZoneDigitalCompareEventCondition
<b>TZCTL</b>	
epwm.h	EPWM_enableTripZoneAdvAction
epwm.h	EPWM_disableTripZoneAdvAction
epwm.h	EPWM_setTripZoneAction
epwm.h	EPWM_setTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB
<b>TZCTL2</b>	
epwm.h	EPWM_enableTripZoneAdvAction
epwm.h	EPWM_disableTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvAction
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>TZCTLDCA</b>	
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionA
<b>TZCTLDCB</b>	
epwm.h	EPWM_setTripZoneAdvDigitalCompareActionB
<b>TZEINT</b>	
epwm.h	EPWM_enableTripZoneInterrupt
epwm.h	EPWM_disableTripZoneInterrupt
<b>TZFLG</b>	
epwm.h	EPWM_getTripZoneFlagStatus
<b>TZCBCFLG</b>	
epwm.h	EPWM_getCycleByCycleTripZoneFlagStatus
<b>TZOSTFLG</b>	
epwm.h	EPWM_getOneShotTripZoneFlagStatus
<b>TZCLR</b>	
epwm.h	EPWM_selectCycleByCycleTripZoneClearEvent
epwm.h	EPWM_clearTripZoneFlag
<b>TZCBCCLR</b>	
epwm.h	EPWM_clearCycleByCycleTripZoneFlag
<b>TZOSTCLR</b>	
epwm.h	EPWM_clearOneShotTripZoneFlag
<b>TZFRC</b>	
epwm.h	EPWM_forceTripZoneEvent
<b>ETSEL</b>	
epwm.h	EPWM_enableInterrupt
epwm.h	EPWM_disableInterrupt
epwm.h	EPWM_setInterruptSource
epwm.h	EPWM_enableADCTrigger
epwm.h	EPWM_disableADCTrigger
epwm.h	EPWM_setADCTriggerSource
<b>ETPS</b>	
epwm.h	EPWM_setInterruptEventCount
epwm.h	EPWM_setADCTriggerEventPrescale
<b>ETFLG</b>	
epwm.h	EPWM_getEventTriggerInterruptStatus
epwm.h	EPWM_getADCTriggerFlagStatus
<b>ETCLR</b>	
epwm.h	EPWM_clearEventTriggerInterruptFlag
epwm.h	EPWM_clearADCTriggerFlag
<b>ETFRC</b>	
epwm.h	EPWM_forceEventTriggerInterrupt
epwm.h	EPWM_forceADCTrigger
<b>ETINTPS</b>	
epwm.h	EPWM_setInterruptEventCount
epwm.h	EPWM_getInterruptEventCount
<b>ETSOCP</b>	

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
epwm.h	EPWM_setADCTriggerEventPrescale
epwm.h	EPWM_getADCTriggerEventCount
<b>ETCNTINITCTL</b>	
epwm.h	EPWM_enableInterruptEventCountInit
epwm.h	EPWM_disableInterruptEventCountInit
epwm.h	EPWM_forceInterruptEventCountInit
epwm.h	EPWM_enableADCTriggerEventCountInit
epwm.h	EPWM_disableADCTriggerEventCountInit
epwm.h	EPWM_forceADCTriggerEventCountInit
<b>ETCNTINIT</b>	
epwm.h	EPWM_enableInterruptEventCountInit
epwm.h	EPWM_disableInterruptEventCountInit
epwm.h	EPWM_forceInterruptEventCountInit
epwm.h	EPWM_setInterruptEventCountInitValue
epwm.h	EPWM_enableADCTriggerEventCountInit
epwm.h	EPWM_disableADCTriggerEventCountInit
epwm.h	EPWM_forceADCTriggerEventCountInit
epwm.h	EPWM_setADCTriggerEventCountInitValue
<b>DCTRIPSEL</b>	
epwm.h	EPWM_selectDigitalCompareTripInput
epwm.h	EPWM_enableDigitalCompareTripCombinationInput
<b>DCACTL</b>	
epwm.h	EPWM_setDigitalCompareEventSource
epwm.h	EPWM_setDigitalCompareEventSyncMode
epwm.h	EPWM_enableDigitalCompareADCTrigger
epwm.h	EPWM_disableDigitalCompareADCTrigger
epwm.h	EPWM_enableDigitalCompareSyncEvent
epwm.h	EPWM_disableDigitalCompareSyncEvent
epwm.h	EPWM_setDigitalCompareCBCLatchMode
epwm.h	EPWM_selectDigitalCompareCBCLatchClearEvent
epwm.h	EPWM_getDigitalCompareCBCLatchStatus
<b>DCBCTL</b>	
-	See DCACTL
<b>DCFCTL</b>	
epwm.h	EPWM_enableDigitalCompareBlankingWindow
epwm.h	EPWM_disableDigitalCompareBlankingWindow
epwm.h	EPWM_enableDigitalCompareWindowInverseMode
epwm.h	EPWM_disableDigitalCompareWindowInverseMode
epwm.h	EPWM_setDigitalCompareBlankingEvent
epwm.h	EPWM_setDigitalCompareFilterInput
epwm.h	EPWM_enableDigitalCompareEdgeFilter
epwm.h	EPWM_disableDigitalCompareEdgeFilter
epwm.h	EPWM_setDigitalCompareEdgeFilterMode
epwm.h	EPWM_setDigitalCompareEdgeFilterEdgeCount
epwm.h	EPWM_getDigitalCompareEdgeFilterEdgeCount

**Table 13-109. EPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
epwm.h	EPWM_getDigitalCompareEdgeFilterEdgeStatus
<b>DCCAPCTL</b>	
epwm.h	EPWM_enableDigitalCompareCounterCapture
epwm.h	EPWM_disableDigitalCompareCounterCapture
epwm.h	EPWM_setDigitalCompareCounterShadowMode
epwm.h	EPWM_getDigitalCompareCaptureStatus
epwm.h	EPWM_clearDigitalCompareCaptureStatusFlag
epwm.h	EPWM_configureDigitalCompareCounterCaptureMode
<b>DCFOFFSET</b>	
epwm.h	EPWM_setDigitalCompareWindowOffset
epwm.h	EPWM_getDigitalCompareBlankingWindowOffsetCount
<b>DCFOFFSETCNT</b>	
epwm.h	EPWM_getDigitalCompareBlankingWindowOffsetCount
<b>DCFWINDOW</b>	
epwm.h	EPWM_setDigitalCompareWindowLength
epwm.h	EPWM_getDigitalCompareBlankingWindowLengthCount
<b>DCFWINDOWCNT</b>	
epwm.h	EPWM_getDigitalCompareBlankingWindowLengthCount
<b>BLANKPULSEMIXSEL</b>	
-	
<b>DCCAP</b>	
epwm.h	EPWM_enableDigitalCompareCounterCapture
epwm.h	EPWM_disableDigitalCompareCounterCapture
epwm.h	EPWM_setDigitalCompareCounterShadowMode
epwm.h	EPWM_getDigitalCompareCaptureStatus
epwm.h	EPWM_clearDigitalCompareCaptureStatusFlag
epwm.h	EPWM_configureDigitalCompareCounterCaptureMode
epwm.h	EPWM_getDigitalCompareCaptureCount
<b>DCAHTRIPSEL</b>	
epwm.h	EPWM_enableDigitalCompareTripCombinationInput
epwm.h	EPWM_disableDigitalCompareTripCombinationInput
<b>DCALTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>DCBHTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>DCBLTRIPSEL</b>	
-	See DCAHTRIPSEL
<b>LOCK</b>	
epwm.h	EPWM_lockRegisters
<b>HWVDELVAL</b>	
epwm.h	EPWM_getValleyHWDelay
<b>VCNTVAL</b>	
epwm.h	EPWM_getValleyCount

**13.17.3.2 HRPWM Registers to Driverlib Functions**
**Table 13-110. HRPWM Registers to Driverlib Functions**

File	Driverlib Function
<b>TBCTL</b>	
-	
<b>TBCTL2</b>	
-	
<b>EPWMSYNCINSEL</b>	
-	
<b>TBCTR</b>	
-	
<b>TBSTS</b>	
-	
<b>EPWMSYNCOUTEN</b>	
-	
<b>TBCTL3</b>	
-	
<b>CMPCTL</b>	
-	
<b>CMPCTL2</b>	
-	
<b>DBCTL</b>	
-	
<b>DBCTL2</b>	
-	
<b>AQCTL</b>	
-	
<b>AQTSRCSEL</b>	
-	
<b>PCCTL</b>	
-	
<b>VCAPCTL</b>	
-	
<b>VCNTCFG</b>	
-	
<b>HRCNFG</b>	
hrpwm.h	HRPWM_setMEPEdgeSelect
hrpwm.h	HRPWM_setMEPControlMode
hrpwm.h	HRPWM_setCounterCompareShadowLoadEvent
hrpwm.h	HRPWM_setOutputSwapMode
hrpwm.h	HRPWM_setChannelBOutputPath
hrpwm.h	HRPWM_enableAutoConversion
hrpwm.h	HRPWM_disableAutoConversion
hrpwm.h	HRPWM_setDeadbandMEPEdgeSelect
hrpwm.h	HRPWM_setRisingEdgeDelayLoadMode
hrpwm.h	HRPWM_setFallingEdgeDelayLoadMode

**Table 13-110. HRPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>HRPWR</b>	
-	
<b>HRMSTEP</b>	
hrpwm.h	HRPWM_setMEPStep
<b>HRCNFG2</b>	
hrpwm.h	HRPWM_setDeadbandMEPEdgeSelect
hrpwm.h	HRPWM_setRisingEdgeDelayLoadMode
hrpwm.h	HRPWM_setFallingEdgeDelayLoadMode
<b>HRPCTL</b>	
hrpwm.h	HRPWM_enablePeriodControl
hrpwm.h	HRPWM_disablePeriodControl
hrpwm.h	HRPWM_enablePhaseShiftLoad
hrpwm.h	HRPWM_disablePhaseShiftLoad
hrpwm.h	HRPWM_setSyncPulseSource
<b>TRREM</b>	
hrpwm.h	HRPWM_setTranslatorRemainder
<b>GLDCTL</b>	
-	
<b>GLDCFG</b>	
-	
<b>EPWMXLINK</b>	
-	
<b>AQCTLA</b>	
-	
<b>AQCTLA2</b>	
-	
<b>AQCTLB</b>	
-	
<b>AQCTLB2</b>	
-	
<b>AQSFRC</b>	
-	
<b>AQCSFRC</b>	
-	
<b>DBREDHR</b>	
hrpwm.h	HRPWM_setRisingEdgeDelay
hrpwm.h	HRPWM_setHiResRisingEdgeDelayOnly
<b>DBRED</b>	
hrpwm.h	HRPWM_setRisingEdgeDelay
hrpwm.h	HRPWM_setHiResRisingEdgeDelayOnly
<b>DBFEDHR</b>	
hrpwm.h	HRPWM_setFallingEdgeDelay
hrpwm.h	HRPWM_setHiResFallingEdgeDelayOnly
<b>DBFED</b>	
hrpwm.h	HRPWM_setFallingEdgeDelay

**Table 13-110. HRPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
hrpwm.h	HRPWM_setHiResFallingEdgeDelayOnly
<b>TBPHS</b>	
hrpwm.h	HRPWM_setPhaseShift
hrpwm.h	HRPWM_setHiResPhaseShiftOnly
<b>TBPRDHR</b>	
hrpwm.h	HRPWM_setTimeBasePeriod
hrpwm.h	HRPWM_setHiResTimeBasePeriodOnly
hrpwm.h	HRPWM_getTimeBasePeriod
hrpwm.h	HRPWM_getHiResTimeBasePeriodOnly
<b>TBPRD</b>	
hrpwm.h	HRPWM_setTimeBasePeriod
hrpwm.h	HRPWM_setHiResTimeBasePeriodOnly
hrpwm.h	HRPWM_getTimeBasePeriod
hrpwm.h	HRPWM_getHiResTimeBasePeriodOnly
<b>CMPA</b>	
hrpwm.h	HRPWM_setCounterCompareValue
hrpwm.h	HRPWM_setHiResCounterCompareValueOnly
hrpwm.h	HRPWM_getCounterCompareValue
hrpwm.h	HRPWM_getHiResCounterCompareValueOnly
<b>CMPB</b>	
hrpwm.h	HRPWM_setCounterCompareValue
hrpwm.h	HRPWM_setHiResCounterCompareValueOnly
hrpwm.h	HRPWM_getCounterCompareValue
hrpwm.h	HRPWM_getHiResCounterCompareValueOnly
<b>CMPC</b>	
-	
<b>CMPD</b>	
-	
<b>GLDCTL2</b>	
-	
<b>SWVDELVAL</b>	
-	
<b>TZSEL</b>	
-	
<b>TZDCSEL</b>	
-	
<b>TZCTL</b>	
-	
<b>TZCTL2</b>	
-	
<b>TZCTLDCA</b>	
-	
<b>TZCTLDCB</b>	
-	
<b>TZEINT</b>	

**Table 13-110. HRPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
-	
<b>TZFLG</b>	
-	
<b>TZCBCFLG</b>	
-	
<b>TZOSTFLG</b>	
-	
<b>TZCLR</b>	
-	
<b>TZCBCCLR</b>	
-	
<b>TZOSTCLR</b>	
-	
<b>TZFRC</b>	
-	
<b>ETSEL</b>	
-	
<b>ETPS</b>	
-	
<b>ETFLG</b>	
-	
<b>ETCLR</b>	
-	
<b>ETFRC</b>	
-	
<b>ETINTPS</b>	
-	
<b>ETSOCPS</b>	
-	
<b>ETCNTINITCTL</b>	
-	
<b>ETCNTINIT</b>	
-	
<b>DCTRIPSEL</b>	
-	
<b>DCACTL</b>	
-	
<b>DCBCTL</b>	
-	
<b>DCFCTL</b>	
-	
<b>DCCAPCTL</b>	
-	
<b>DCFOFFSET</b>	
-	

**Table 13-110. HRPWM Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>DCOFFSETCNT</b>	
-	
<b>DCFWINDOW</b>	
-	
<b>DCFWINDOWCNT</b>	
-	
<b>BLANKPULSEMIXSEL</b>	
-	
<b>DCCAP</b>	
-	
<b>DCAHTRIPSEL</b>	
-	
<b>DCALTRIPSEL</b>	
-	
<b>DCBHTRIPSEL</b>	
-	
<b>DCBLTRIPSEL</b>	
-	
<b>EPWMLOCK</b>	
hrpwm.h	HRPWM_lockRegisters
<b>HWVDELVAL</b>	
-	
<b>VCNTVAL</b>	
-	

## Chapter 14

# Enhanced Capture (eCAP)

---



This chapter describes the enhanced capture (eCAP) module, which is used in systems where accurate timing of external events is important.

The enhanced capture (eCAP) module is a Type 2 eCAP. See the [C2000 Real-Time Control Peripheral Reference Guide](#) for a list of all devices with an eCAP module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

<b>14.1 Introduction</b> .....	<b>1516</b>
<b>14.2 Description</b> .....	<b>1517</b>
<b>14.3 Configuring Device Pins for the eCAP</b> .....	<b>1517</b>
<b>14.4 Capture and APWM Operating Mode</b> .....	<b>1520</b>
<b>14.5 Capture Mode Description</b> .....	<b>1522</b>
<b>14.6 Application of the eCAP Module</b> .....	<b>1530</b>
<b>14.7 Application of the APWM Mode</b> .....	<b>1534</b>
<b>14.8 Software</b> .....	<b>1535</b>
<b>14.9 eCAP Registers</b> .....	<b>1535</b>

## 14.1 Introduction

### 14.1.1 Features

The features of the eCAP module include:

- Speed measurements of rotating machinery (for example, toothed sprockets sensed by way of 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 features described in this chapter include:

- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single-shot capture of up to four event time-stamps
- Continuous mode capture of time stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- When not used in capture mode, the eCAP module can be configured as a single-channel PWM output

The capture functionality of the Type 1 eCAP is enhanced from the Type 0 eCAP with the following added features:

- Event filter reset bit
  - Writing a 1 to ECCTL2[CTRFILTRESET] clears the event filter, the modulo counter, and any pending interrupts flags. Resetting the bit is useful for initialization and debug.
- Modulo counter status bits
  - The modulo counter (ECCTL2 [MODCNRSTS]) indicates which capture register is loaded next. In the Type 0 eCAP, to know the current state of the modulo counter was not possible
- Input multiplexer
  - ECCTL0 [INPUTSEL] selects one of 128 input signals, which are detailed in [Section 14.3](#).
- EALLOW protection
  - EALLOW protection was added to critical registers. To maintain software compatibility with Type-0, configure DEV\_CFG\_REGS.ECAPTYPE to make these registers unprotected.

The capture functionality of the Type 2 eCAP is enhanced from the Type 1 eCAP with the following added features:

- Added ECAPxSYNCINSEL register
  - ECAPxSYNCINSEL register is added for each eCAP to select an external SYNCIN. Every eCAP can have a separate SYNCIN signal.

### 14.1.2 ECAP Related Collateral

#### Foundational Materials

- [C2000 Academy - ECAP](#)

#### Getting Started Materials

- [Leveraging High Resolution Capture \(HRCAP\) for Single Wire Data Transfer Application Report](#)

## 14.2 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:

- Capture inputs can be connected using the Input X-BAR
- 128:1 input multiplexer
- Output X-BAR is used to configure output in APWM mode
- 32-bit time base (counter)
- 4 x 32-bit time-stamp capture registers (CAP1-CAP4)
- Four-stage sequencer (modulo4 counter) that is synchronized to external events, eCAP pin rising/falling edges.
- Modulo counter status register (MODCNRSTS) to indicate sequencer state
- Independent edge polarity (rising/falling edge) selection for all four events
- Input capture signal prescaling (from 2-62 or bypass)
- One-shot compare register (two bits) to freeze captures after 1-4 time-stamp events
- Control for continuous time-stamp captures using a four-deep circular buffer (CAP1-CAP4) scheme
- Ability to reset event filter, modulo counter, and interrupt flags
- Interrupt capabilities on any of the four capture events
- EALLOW protection to control registers

## 14.3 Configuring Device Pins for the eCAP

The Input X-BAR connects the device pins to the module as input. Any GPIO on the device can be configured as an input. The GPIO input qualification can be set to synchronous or asynchronous mode by setting the GPxQSELn register bits. Using synchronized inputs can help with noise immunity but affects the eCAP accuracy by  $\pm 2$  cycles. The internal pull-ups can be configured in the GPyPUD register. Since the GPIO mode is used, the GPyINV register can invert the signals.

New to the Type 1 eCAP module, a 128:1 input multiplexer must also be configured (see [Figure 14-3](#)). This multiplexer can select a variety of inputs detailed in [Table 14-1](#) by configuring ECCTL0.INPUTSEL.

**Table 14-1. eCAP Input Selection**

Selection of eCAP Input	ECAP1 INDEX	ECAP2 INDEX
INPUTXBAR1	0	0
INPUTXBAR2	1	1
INPUTXBAR3	2	2
INPUTXBAR4	3	3
INPUTXBAR5	4	4
INPUTXBAR6	5	5
INPUTXBAR7	6	6
INPUTXBAR8	7	7
INPUTXBAR9	8	8
INPUTXBAR10	9	9
INPUTXBAR11	10	10
INPUTXBAR12	11	11
INPUTXBAR13	12	12
INPUTXBAR14	13	13
INPUTXBAR15	14	14
INPUTXBAR16	15	15
Reserved	16-19	16-19
CANA_INT0	20	20
Reserved	21-23	21-23
OUTPUTXBAR1	24	24
OUTPUTXBAR2	25	25
OUTPUTXBAR3	26	26
OUTPUTXBAR4	27	27
OUTPUTXBAR5	28	28
OUTPUTXBAR6	29	29
OUTPUTXBAR7	30	30
OUTPUTXBAR8	31	31
Reserved	32-35	32-35
ADCCEVT1	36	36
ADCCEVT2	37	37
ADCCEVT3	38	38
ADCCEVT4	39	39
Reserved	40-43	40-43
ADCAEVT1	44	44
ADCAEVT2	45	45
ADCAEVT3	46	46
ADCAEVT4	47	47
Reserved	48-95	48-95

**Table 14-1. eCAP Input Selection (continued)**

Selection of ECAP Input	ECAP1 INDEX	ECAP2 INDEX
CMPSS1_CTRIPL	96	96
CMPSS2_CTRIPL	97	97
CMPSS3_CTRIPL	98	98
CMPSS4_CTRIPL	99	99
Reserved	100-107	100-107
CMPSS1_CTRIPH	108	108
CMPSS2_CTRIPH	109	109
CMPSS3_CTRIPH	110	110
CMPSS4_CTRIPH	111	111
Reserved	112-114	112-114
GPIO8	115	115
GPIO9	116	116
GPIO22	117	117
GPIO23	118	118
Reserved	119	119
CMPSS1_CTRIPH_OR_CTRIPL	120	120
CMPSS2_CTRIPH_OR_CTRIPL	121	121
CMPSS3_CTRIPH_OR_CTRIPL	122	122
CMPSS4_CTRIPH_OR_CTRIPL	123	123
Reserved	124-126	124-126
INPUTXBAR7	127	Reserved
INPUTXBAR8	Reserved	127
INPUTXBAR9	Reserved	Reserved

The Output X-BAR must be used to connect output signals to the OUTPUTXBARx output locations. The GPIO mux must then be configured to connect the OUTPUTXBARx lines to any of several IO pins with the GPIO mux. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

See the *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux, GPIO settings, and XBAR configuration.

---

**Note**

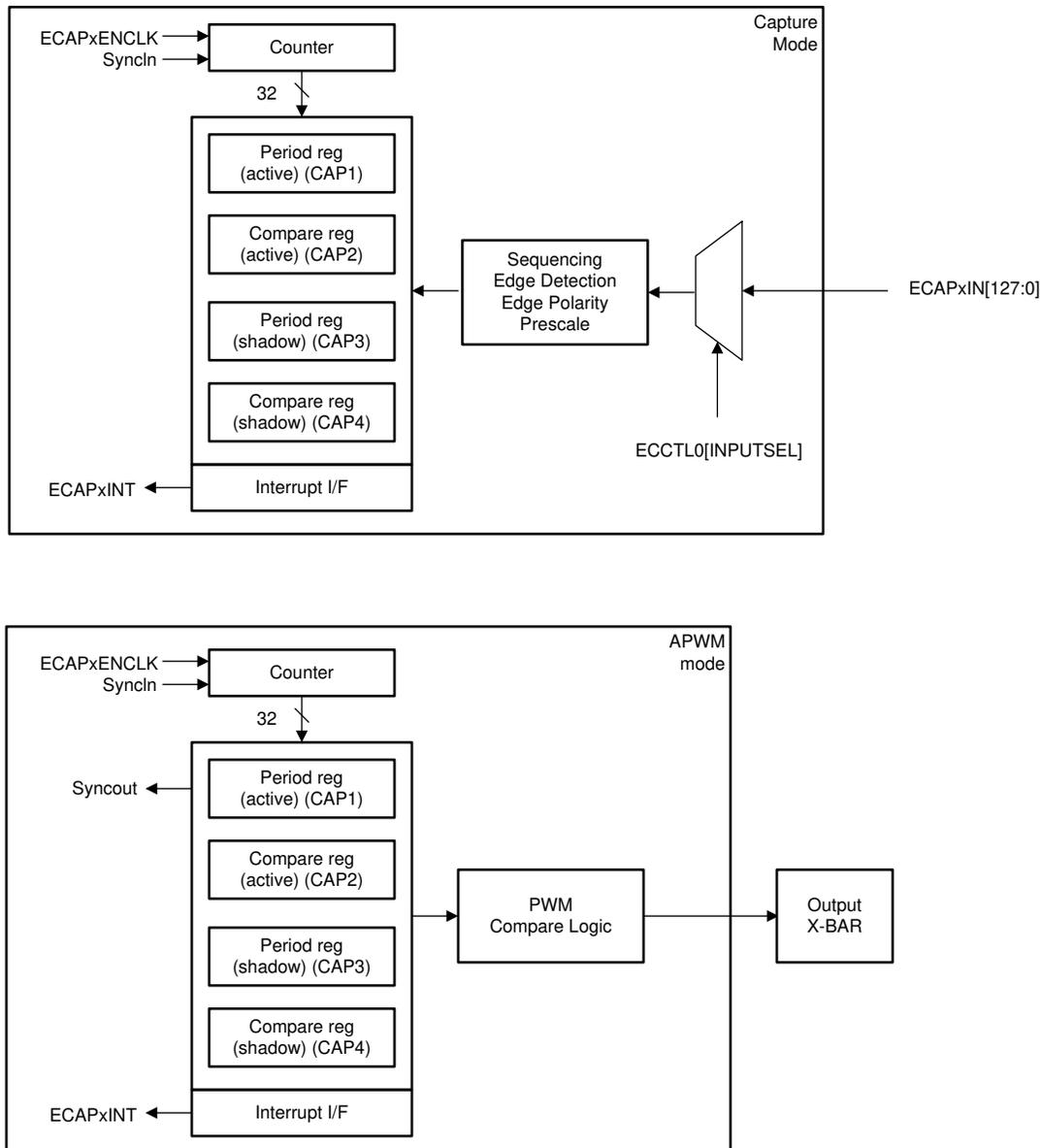
ECAPxIN has to be at least  $2 \times \text{SYSCLK}$ -cycles wide to be properly captured by the eCAP module; otherwise, the input pulse can get missed from sampling by the SYSCLK.

---

## 14.4 Capture and APWM Operating Mode

Use the eCAP module resources to implement a single-channel PWM generator (with 32-bit capabilities) when the eCAP module 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 compare shadow registers, respectively. Figure 14-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 14-2 further describes the output of the eCAP in APWM mode based on the CMP and PRD values.



- A single pin is shared between CAP and APWM functions. In capture mode, the pin is an input; in APWM mode, the pin is an output.
- 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 14-1. Capture and APWM Modes of Operation**

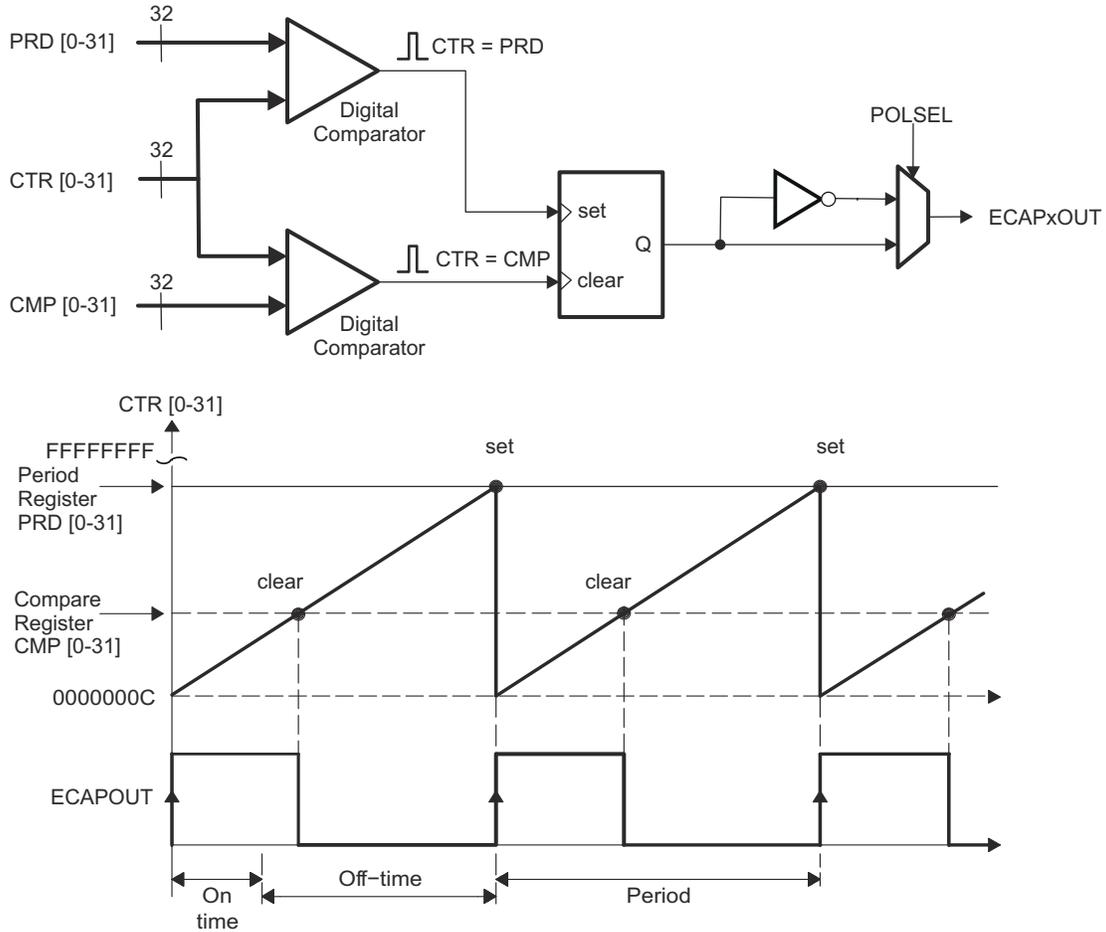
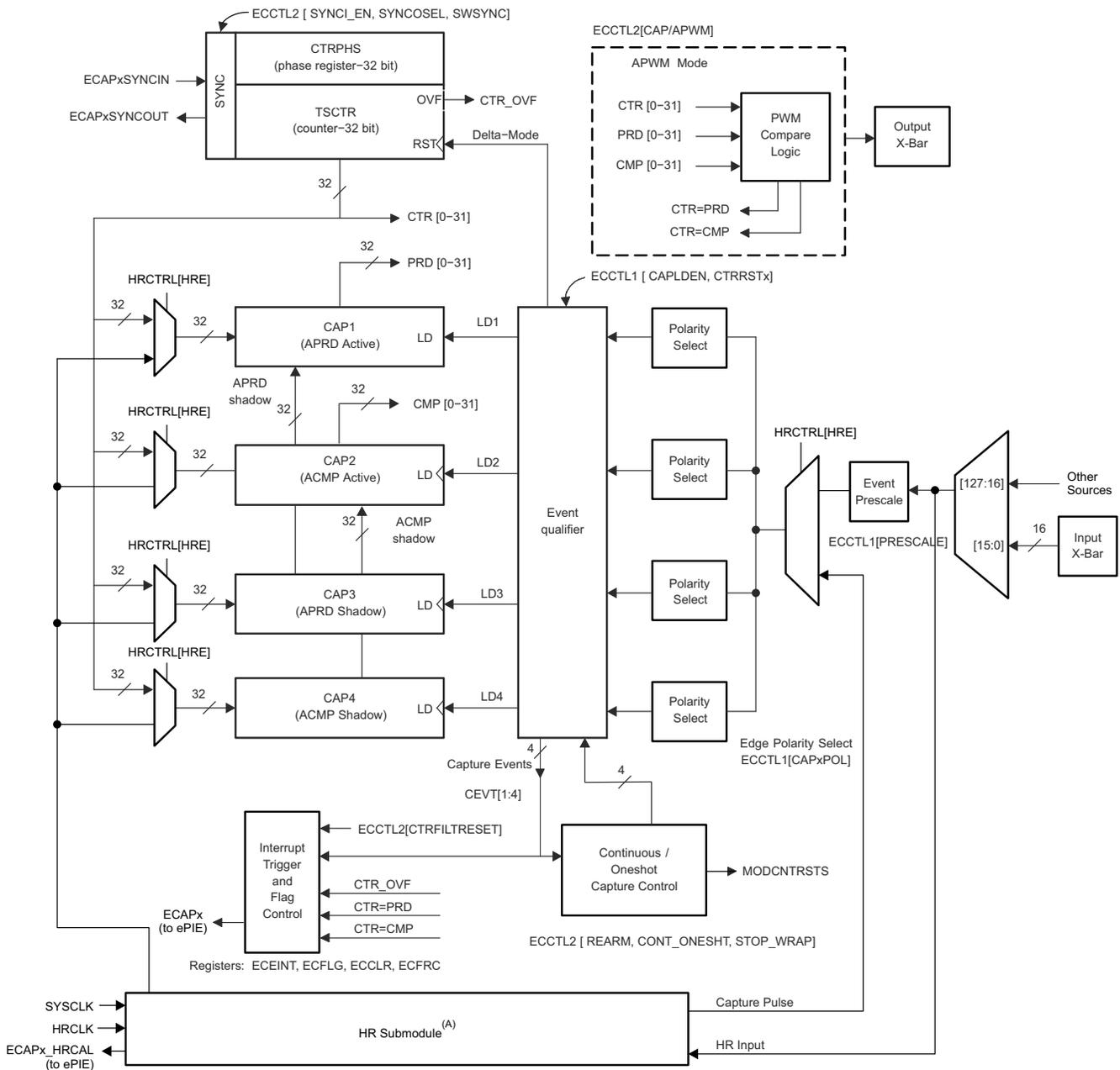


Figure 14-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode

### 14.5 Capture Mode Description

Figure 14-3 shows the various components that implement the capture function.

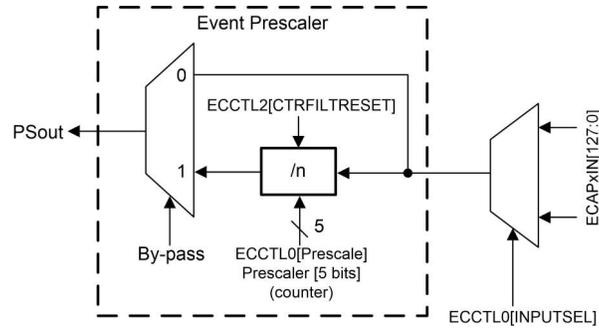


A. The HRCAP submodule is not available on all eCAP modules; in this case, the high-resolution muxes and hardware are not implemented.

Figure 14-3. eCAP Block Diagram

### 14.5.1 Event Prescaler

An input capture signal (pulse train) can be prescaled by  $N = 2-62$  (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. Figure 14-4 shows a functional diagram and Figure 14-5 shows the operation of the prescale function. The event prescaler can be reset by setting the ECCTL2.CTRFILTRESET register bit.



- A. When a prescale value of 1 is chosen (ECCTL1[13:9] = 0,0,0,0,0), the input capture signal bypasses the prescale logic completely.
- B. The first Rise edge after Prescale configuration change is not passed to Capture logic, prescaler value takes into effect on the second rising edge after the configuration.

Figure 14-4. Event Prescale Control

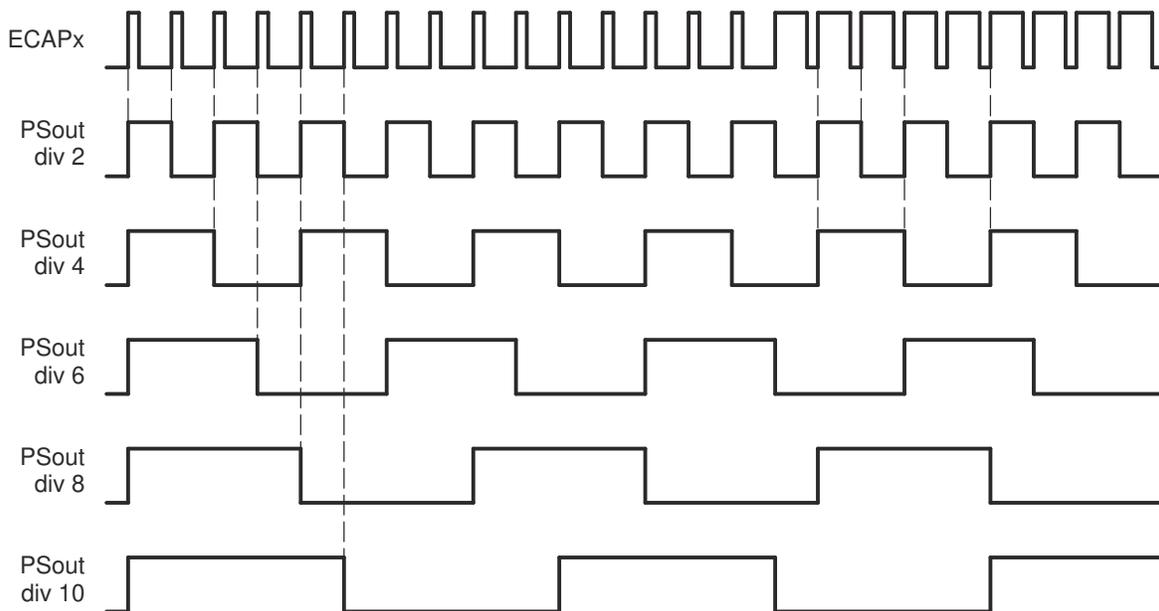


Figure 14-5. Prescale Function Waveforms

### 14.5.2 Edge Polarity Select and Qualifier

Functionality and features include:

- Four independent edge polarity (rising edge/falling edge) selection muxes are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to the respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

### 14.5.3 Continuous/One-Shot Control

Operation of eCAP in Continuous/One-Shot mode:

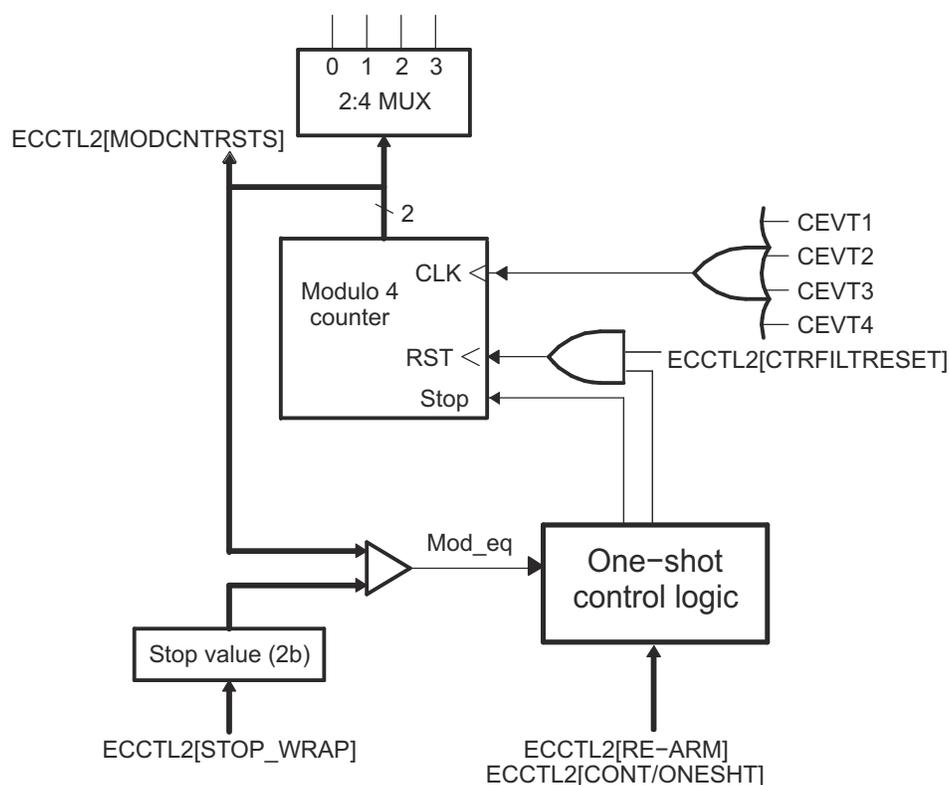
- The Mod4 (2-bit) counter is incremented using edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- During one-shot operation, a 2-bit stop register (STOP\_WRAP) is used to compare the Mod4 counter output, and when equal, stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. In this mode, if TSCCTR counter is configured to reset on capture event (CEVTx) by configuring ECCTL1.CTRRSTx bit, the operation still keeps resetting the TSCCTR counter on capture event (CEVTx) after the STOP\_WRAP value is reached and re-arm (REARM) has not occurred.

The continuous/one-shot block controls the start, stop and reset (zero) functions of the Mod4 counter, using a mono-shot type of action that can be triggered by the stop-value comparator and re-armed using 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 14-6. Details of the Continuous/One-shot Block**

### 14.5.4 32-Bit Counter and Phase Control

This counter provides the time-base for event captures, and is clocked using the system clock.

A phase register is provided to achieve synchronization with other counters using 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 the counter value is reset to 0 by any of the LD1-LD4 signals.

### 14.5.5 CAP1-CAP4 Registers

These 32-bit registers are supplied by the 32-bit counter timer bus, CTR[0-31], and are loaded (capture a time-stamp) when the respective LD inputs are strobed.

Control bit CAPLDEN can inhibit loading of the capture registers. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

### 14.5.6 eCAP Synchronization

eCAP modules can be synchronized with each other by selecting a common SYNCIN source. SYNCIN source for eCAP can be either software sync-in or external sync-in. The external sync-in signal can come from eCAP, X-Bar or EPWM. The SWSYNC of the eCAP module is logical ORed with the SYNC signal as shown in Figure 14-7. The SYNC signal is defined by the selection of ECAPxSYNCINSEL[SEL] as shown in Figure 14-8.

#### Note

ECAPxSYNCOUT going to the ECAPSYNCIN multiplexer is disabled. For example, ECAP1SYNCOUT cannot be a sync in to ECAP1, but ECAP1SYNCOUT can be a sync in to ECAP2, ECAP3, and so on.

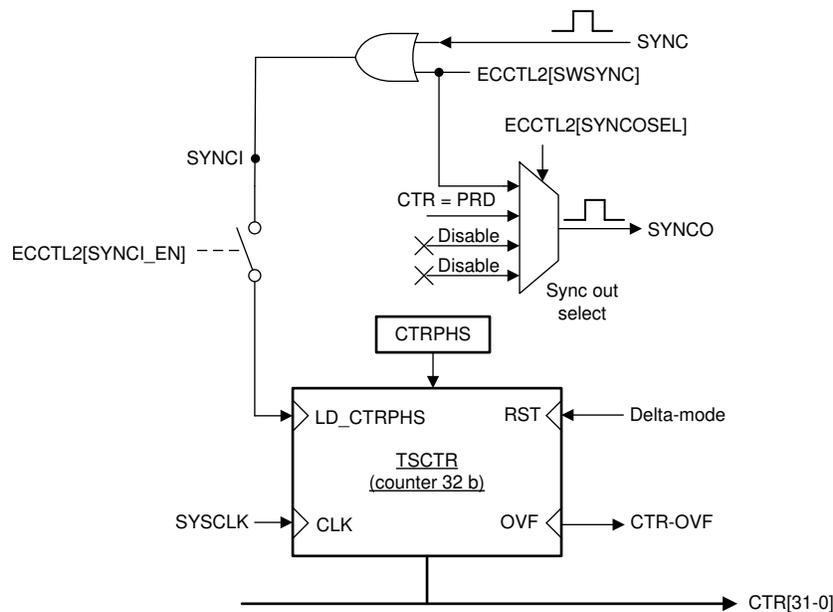
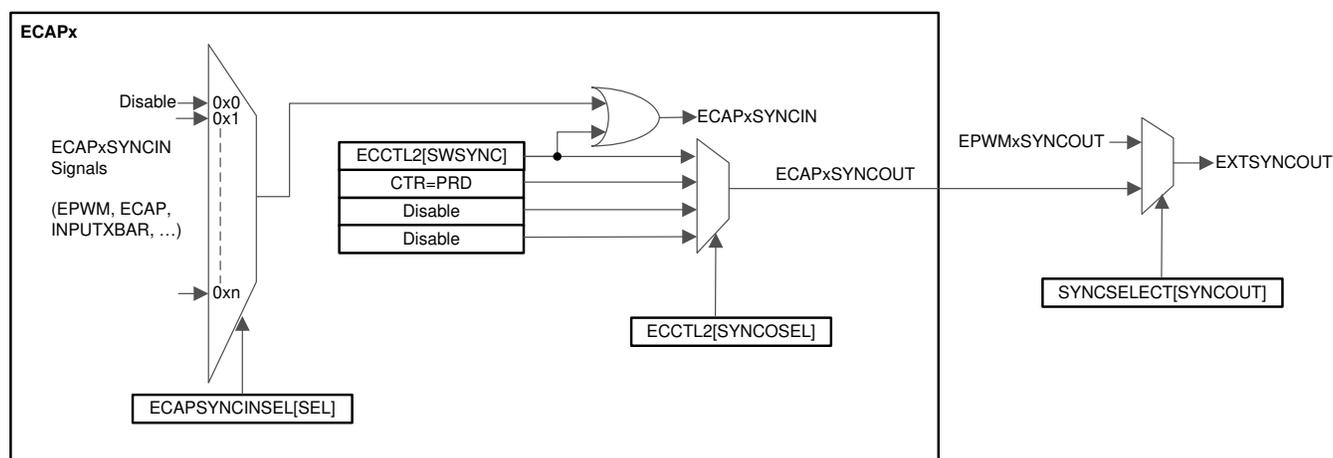


Figure 14-7. Details of the Counter and Synchronization Block



**Figure 14-8. eCAP Synchronization Scheme**

#### 14.5.6.1 Example 1 - Using SWSYNC with ECAP Module

Implement the following steps to use SWSYNC with ECAP1 and ECAP2.

- Configure ECAP[1..2].ECAPSYNCINSEL.SEL = 0x0 to disable external SYNCIN coming to eCAP1.
- Configure ECAP[1..2].ECCTL2.SWSYNC = 0x1, to force Software Synchronization of the TSCTR counter.

To use SWSYNC with other eCAP modules, make sure that the previous eCAP chain is not generating a SYNCOUT signal that interferes with the software synchronization.

#### 14.5.7 Interrupt Control

Operation and features of the eCAP interrupt control include (see [Figure 14-9](#)):

- An interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP).
- A counter overflow event (FFFFFFFF->00000000) is also provided as an interrupt source (CTROVF).
- The capture events are edge and sequencer-qualified (ordered in time) by the polarity select and Mod4 gating, respectively.
- One of these events can be selected as the interrupt source (from the eCAPx module) going to the PIE.
- Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=COMP) can be generated.
- The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event using the interrupt clear register (ECCLR) before any other interrupt pulses are generated. All interrupt flags are cleared upon an event filter reset by writing a 1 to ECCTL2[CLRFILTRESET]. To force an interrupt event, use the interrupt force register (ECFRC). This is useful for test purposes.

#### Note

The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR=PRD, CTR=COMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

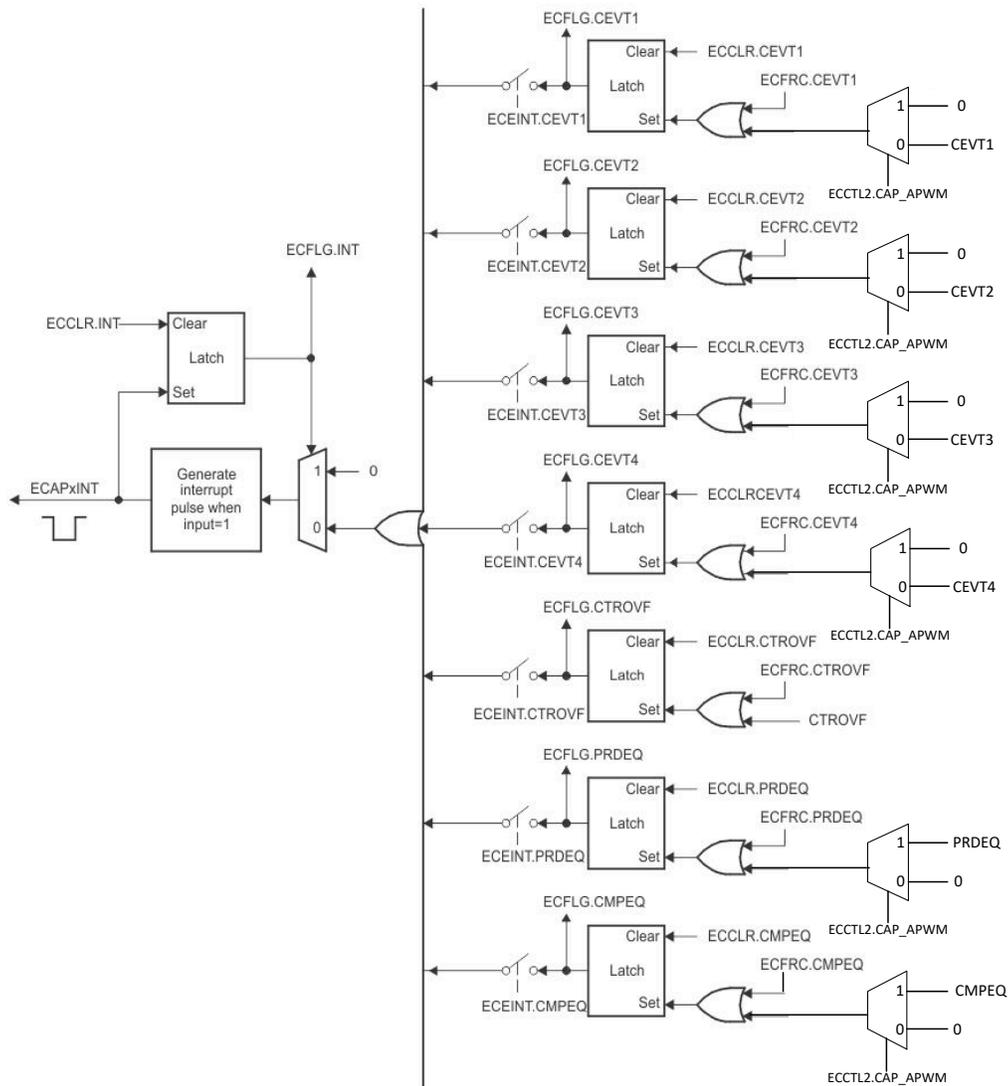


Figure 14-9. Interrupts in eCAP Module

### 14.5.8 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.

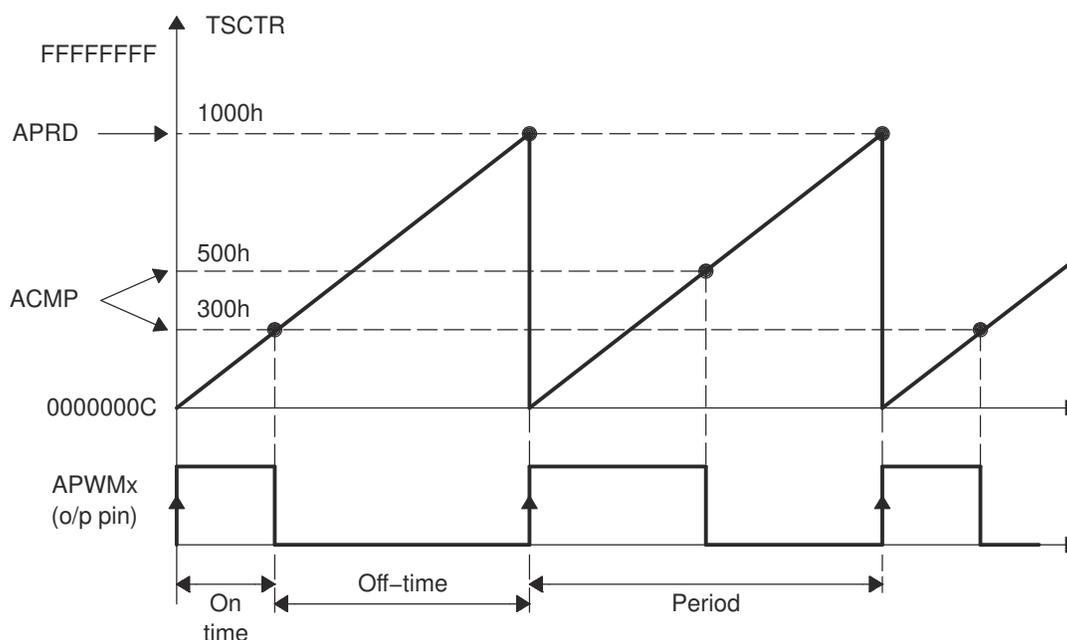
In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal,  $CTR[31:0] = PRD[31:0]$ .

### 14.5.9 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison by way of 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, the contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved using 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 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.
- During initialization, 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, use the shadow registers.



**Figure 14-10. PWM Waveform Details Of APWM Mode Operation**

The behavior of APWM active high mode (APWMPOL == 0) is as follows:

CMP = 0x00000000, output low for duration of period (0% duty)

CMP = 0x00000001, output high 1 cycle

CMP = 0x00000002, output high 2 cycles

CMP = PERIOD, output high except for 1 cycle (<100% duty)

CMP = PERIOD+1, output high for complete period (100% duty)

CMP > PERIOD+1, output high for complete period

The behavior of APWM active low mode (APWMPOL == 1) is as follows:

CMP = 0x00000000, output high for duration of period (0% duty)

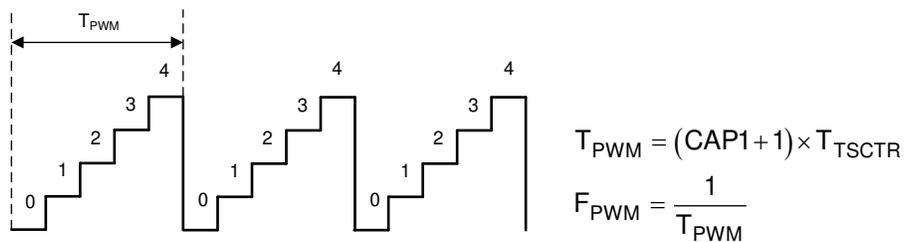
CMP = 0x00000001, output low 1 cycle

CMP = 0x00000002, output low 2 cycles

CMP = PERIOD, output low except for 1 cycle (<100% duty)

CMP = PERIOD+1, output low for complete period (100% duty)

CMP > PERIOD+1, output low for complete period



**Figure 14-11. Time-Base Frequency and Period Calculation**

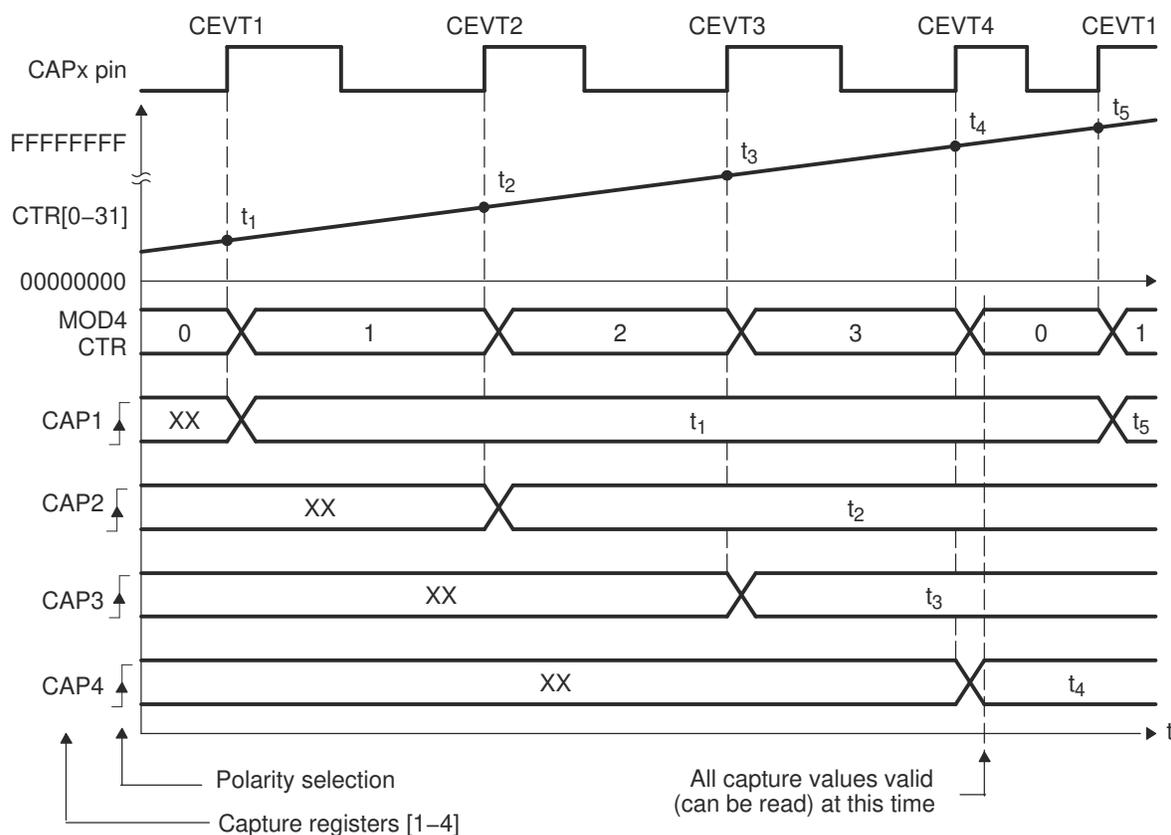
## 14.6 Application of the eCAP Module

The following sections provide applications examples to show how to operate the eCAP module.

### 14.6.1 Example 1 - Absolute Time-Stamp Operation Rising-Edge Trigger

Figure 14-12 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFFFFFF (maximum value), the Mod4 counter wraps around to 00000000 (not shown in Figure 14-12), if this occurs, the 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 fourth event); hence, event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPx registers.



**Figure 14-12. Capture Sequence for Absolute Time-stamp and Rising-Edge Detect**

### 14.6.2 Example 2 - Absolute Time-Stamp Operation Rising- and Falling-Edge Trigger

In Figure 14-13, the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information, that is:  $\text{Period1} = t_3 - t_1$ ,  $\text{Period2} = t_5 - t_3$ , ...and so on.  $\text{Duty Cycle1 (on-time \%)} = (t_2 - t_1) / \text{Period1} \times 100\%$ , and so on.  $\text{Duty Cycle1 (off-time \%)} = (t_3 - t_2) / \text{Period1} \times 100\%$ , and so on.

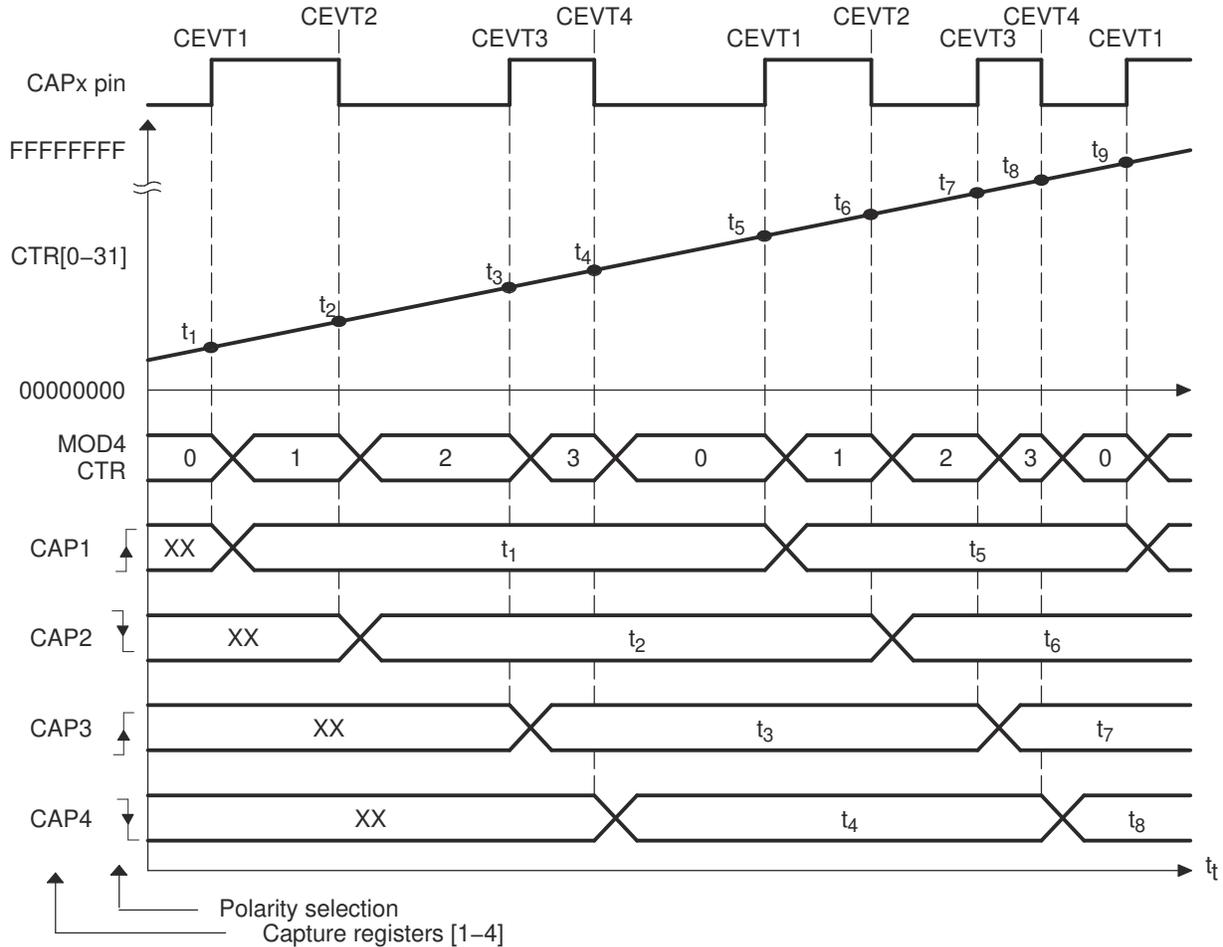


Figure 14-13. Capture Sequence for Absolute Time-stamp with Rising- and Falling-Edge Detect

### 14.6.3 Example 3 - Time Difference (Delta) Operation Rising-Edge Trigger

Figure 14-14 shows how the eCAP module can be used to collect delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is reset back to zero on every valid event. Here capture events are qualified as rising edge only. On an event, TSCTR contents (Time-Stamp) is captured first, and then TSCTR is reset to zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFFFFFF (maximum value), before the next event, the Mod4 counter wraps around to 00000000 and continues, a CNTOVF (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 =  $T_2$ , and so on. As shown in Figure 14-14, 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.

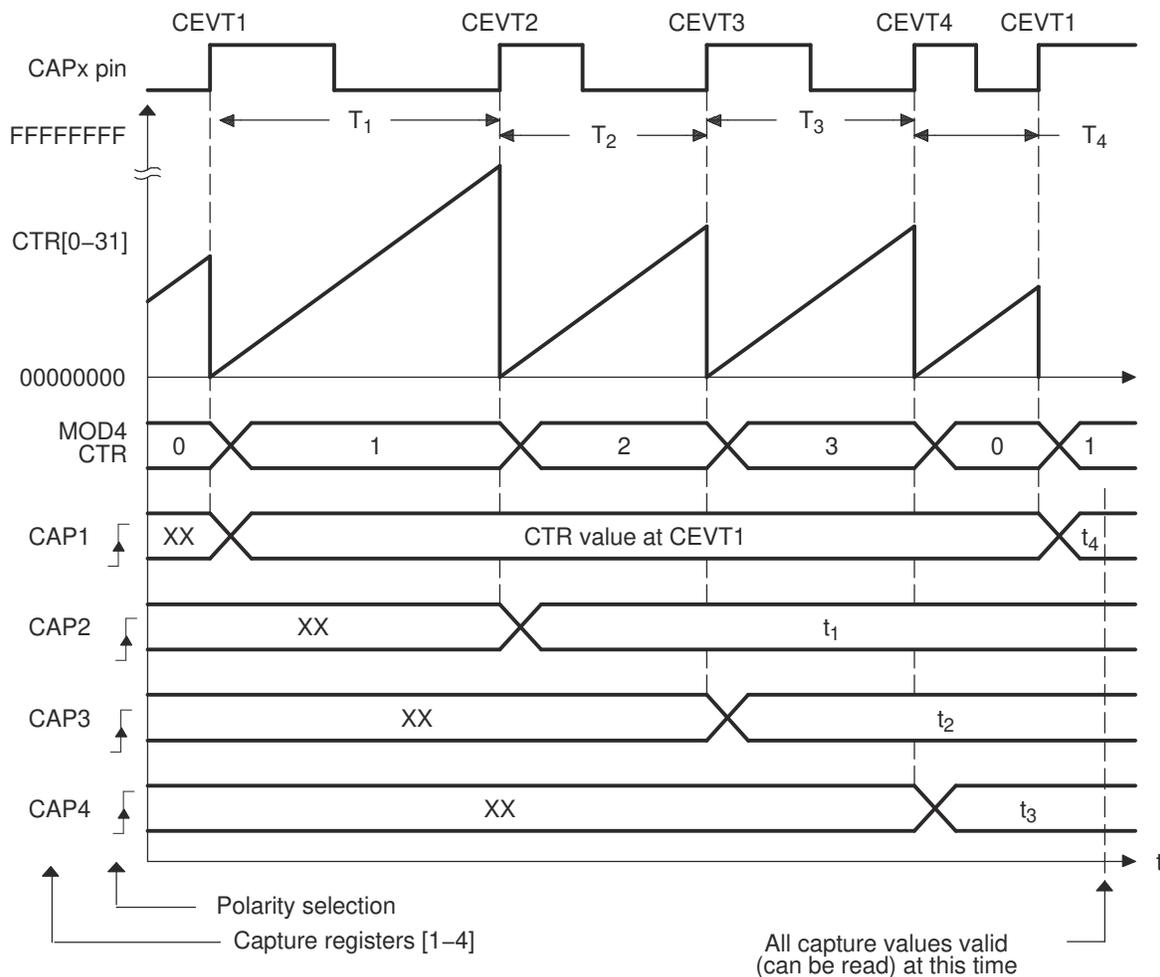


Figure 14-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect

### 14.6.4 Example 4 - Time Difference (Delta) Operation Rising- and Falling-Edge Trigger

In Figure 14-15, the eCAP operating mode is almost the same as in previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information, that is: Period1 =  $T_1+T_2$ , Period2 =  $T_3+T_4$ , and so on. Duty Cycle1 (on-time %) =  $T_1 / \text{Period1} \times 100\%$ , Duty Cycle1 (off-time %) =  $T_2 / \text{Period1} \times 100\%$ , and so on.

During initialization, write to the active registers for both period and compare. This action automatically copies the init values into the shadow values. For subsequent compare updates during run-time, the shadow registers must be used.

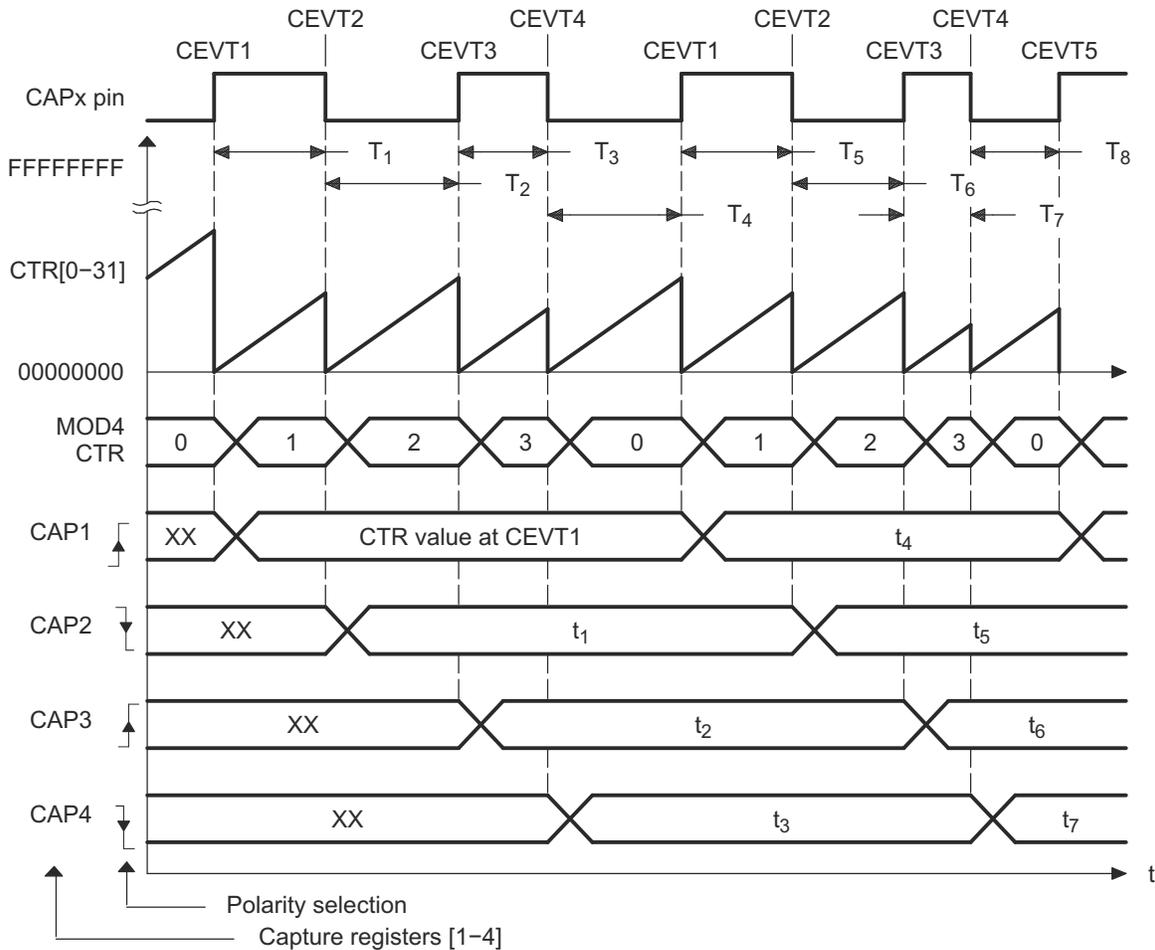
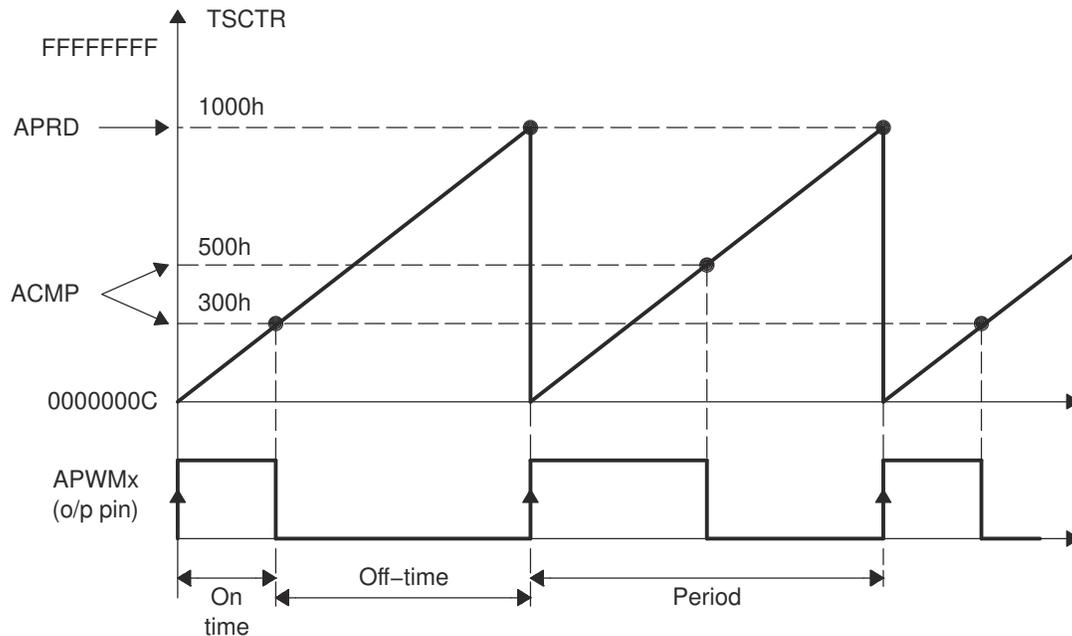


Figure 14-15. Capture Sequence for Delta Mode Time-stamp with Rising- and Falling-Edge Detect

## 14.7 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 the APWMx output pin. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.

### 14.7.1 Example 1 - Simple PWM Generation (Independent Channels)



**Figure 14-16. PWM Waveform Details of APWM Mode Operation**

## 14.8 Software

### 14.8.1 ECAP Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/ecap

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](#).

#### 14.8.1.1 eCAP APWM Example

FILE: `ecap_ex1_apwm.c`

This program sets up the eCAP module in APWM mode. The PWM waveform will come out on GPIO5. The frequency of PWM is configured to vary between 5Hz and 10Hz using the shadow registers to load the next period/compare values.

#### 14.8.1.2 eCAP Capture PWM Example

FILE: `ecap_ex2_capture_pwm.c`

This example configures ePWM3A for:

- Up count mode
- Period starts at 500 and goes up to 8000
- Toggle output on PRD

eCAP1 is configured to capture the time between rising and falling edge of the ePWM3A output.

#### External Connections

- eCAP1 is on GPIO16
- ePWM3A is on GPIO4
- Connect GPIO4 to GPIO16.

#### Watch Variables

- `ecap1PassCount` - Successful captures.
- `ecap1IntCount` - Interrupt counts.

#### 14.8.1.3 eCAP APWM Phase-shift Example

FILE: `ecap_ex3_apwm_phase_shift.c`

This program sets up the eCAP1 and eCAP2 modules in APWM mode to generate the two phase-shifted PWM outputs of same duty and frequency value. The frequency, duty and phase values can be programmed of choice by updating the defined macros. By default 10 Khz frequency, 50% duty and 30% phase shift values are used. eCAP2 output leads the eCAP1 output by 30%. GPIO5 and GPIO6 are used as eCAP1/2 outputs and can be probed using analyzer/CRO to observe the waveforms.

## 14.9 eCAP Registers

This section describes the Enhanced Capture Registers.

### 14.9.1 ECAP Base Address Table

**Table 14-2. ECAP Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
ECap1Regs	ECAP_REGS	ECAP1_BASE	0x0000_5200	YES
ECap2Regs	ECAP_REGS	ECAP2_BASE	0x0000_5240	YES

## 14.9.2 ECAP\_REGS Registers

Table 14-3 lists the memory-mapped registers for the ECAP\_REGS registers. All register offset addresses not listed in Table 14-3 should be considered as reserved locations and the register contents should not be modified.

**Table 14-3. ECAP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	TSCTR	Time-Stamp Counter		<a href="#">Go</a>
2h	CTRPHS	Counter Phase Offset Value Register		<a href="#">Go</a>
4h	CAP1	Capture 1 Register		<a href="#">Go</a>
6h	CAP2	Capture 2 Register		<a href="#">Go</a>
8h	CAP3	Capture 3 Register		<a href="#">Go</a>
Ah	CAP4	Capture 4 Register		<a href="#">Go</a>
12h	ECCTL0	Capture Control Register 0	EALLOW	<a href="#">Go</a>
14h	ECCTL1	Capture Control Register 1	EALLOW	<a href="#">Go</a>
15h	ECCTL2	Capture Control Register 2	EALLOW	<a href="#">Go</a>
16h	ECEINT	Capture Interrupt Enable Register	EALLOW	<a href="#">Go</a>
17h	ECFLG	Capture Interrupt Flag Register		<a href="#">Go</a>
18h	ECCLR	Capture Interrupt Clear Register		<a href="#">Go</a>
19h	ECFRC	Capture Interrupt Force Register	EALLOW	<a href="#">Go</a>
1Eh	ECAPSYNCINSEL	SYNC source select register	EALLOW	<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 14-4 shows the codes that are used for access types in this section.

**Table 14-4. ECAP\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value

### 14.9.2.1 TSCTR Register (Offset = 0h) [Reset = 0000000h]

TSCTR is shown in [Figure 14-17](#) and described in [Table 14-5](#).

Return to the [Summary Table](#).

Time-Stamp Counter

**Figure 14-17. TSCTR Register**

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

**Table 14-5. TSCTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TSCTR	R/W	0h	Active 32-bit counter register that is used as the capture time-base HR mode : 1) This register reads HRCOUNTER value and is not writable 2) can be reset using CTRFILTRRESET 3) Its not synchronized to SYSCLK domain so reads may not be accurate Reset type: SYSRSn

### 14.9.2.2 CTRPHS Register (Offset = 2h) [Reset = 0000000h]

CTRPHS is shown in [Figure 14-18](#) and described in [Table 14-6](#).

Return to the [Summary Table](#).

Counter Phase Offset Value Register

**Figure 14-18. CTRPHS Register**

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

**Table 14-6. 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 CTRPHS is loaded into TSCTR upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases. This register is not applicable in HR mode. Reset type: SYSRSn

### 14.9.2.3 CAP1 Register (Offset = 4h) [Reset = 00000000h]

CAP1 is shown in [Figure 14-19](#) and described in [Table 14-7](#).

Return to the [Summary Table](#).

Capture 1 Register

**Figure 14-19. CAP1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP1																															
R/W-0h																															

**Table 14-7. CAP1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP1	R/W	0h	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>- Time-Stamp counter value (TSCTR) during a capture event</li> <li>- Software - may be useful for test purposes or initialization</li> <li>- ARPD shadow register (CAP3) when used in APWM mode</li> </ul> Reset type: SYSRSn

#### 14.9.2.4 CAP2 Register (Offset = 6h) [Reset = 0000000h]

CAP2 is shown in [Figure 14-20](#) and described in [Table 14-8](#).

Return to the [Summary Table](#).

Capture 2 Register

**Figure 14-20. CAP2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP2																															
R/W-0h																															

**Table 14-8. CAP2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP2	R/W	0h	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>- Time-Stamp ( counter value) during a capture event</li> <li>- Software - may be useful for test purposes</li> <li>- ACMP shadow register (CAP4) when used in APWM mode</li> </ul> Reset type: SYSRSn

### 14.9.2.5 CAP3 Register (Offset = 8h) [Reset = 0000000h]

CAP3 is shown in [Figure 14-21](#) and described in [Table 14-9](#).

Return to the [Summary Table](#).

Capture 3 Register

**Figure 14-21. CAP3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP3																															
R/W-0h																															

**Table 14-9. CAP3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CAP3	R/W	0h	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. You can update the PWM period value through this register. CAP3 (APRD) shadows CAP1 in this mode. Reset type: SYSRSn

### 14.9.2.6 CAP4 Register (Offset = Ah) [Reset = 0000000h]

CAP4 is shown in [Figure 14-22](#) and described in [Table 14-10](#).

Return to the [Summary Table](#).

Capture 4 Register

**Figure 14-22. CAP4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAP4																															
R/W-0h																															

**Table 14-10. 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 can update the PWM compare value via this register. CAP4 (ACMP) shadows CAP2 in this mode. Reset type: SYSRSn

### 14.9.2.7 ECCTL0 Register (Offset = 12h) [Reset = 000007Fh]

ECCTL0 is shown in [Figure 14-23](#) and described in [Table 14-11](#).

Return to the [Summary Table](#).

Capture Control Register 0

**Figure 14-23. ECCTL0 Register**

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

**Table 14-11. ECCTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-7	RESERVED	R-0	0h	Reserved
6-0	INPUTSEL	R/W	7Fh	Capture input source select bits 0000000 capture input is ECAPxINPUT[0] 0000001 capture input is ECAPxINPUT[1] 0000010 capture input is ECAPxINPUT[2] ... 1111111 capture input is ECAPxINPUT[127] Reset type: CPU1.SYSRSn

### 14.9.2.8 ECCTL1 Register (Offset = 14h) [Reset = 0000h]

ECCTL1 is shown in [Figure 14-24](#) and described in [Table 14-12](#).

Return to the [Summary Table](#).

Capture Control Register 1

**Figure 14-24. ECCTL1 Register**

15		14		13		12		11		10		9		8	
FREE_SOFT				PRESCALE								CAPLDEN			
R/W-0h				R/W-0h								R/W-0h			
7		6		5		4		3		2		1		0	
CTRRST4		CAP4POL		CTRRST3		CAP3POL		CTRRST2		CAP2POL		CTRRST1		CAP1POL	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 14-12. ECCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Control Reset type: SYSRSn 0h (R/W) = TSCTR counter stops immediately on emulation suspend 1h (R/W) = TSCTR counter runs until = 0 2h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free) 3h (R/W) = TSCTR counter is unaffected by emulation suspend (Run Free)
13-9	PRESCALE	R/W	0h	Event Filter prescale select Reset type: SYSRSn 0h (R/W) = Divide by 1 (i.e., no prescale, by-pass the prescaler) 1h (R/W) = Divide by 2 2h (R/W) = Divide by 4 3h (R/W) = Divide by 6 4h (R/W) = Divide by 8 5h (R/W) = Divide by 10 1Eh (R/W) = Divide by 60 1Fh (R/W) = Divide by 62
8	CAPLDEN	R/W	0h	Enable Loading of CAP1-4 registers on a capture event. Note that this bit does not disable CEVTn events from being generated. Reset type: SYSRSn 0h (R/W) = Disable CAP1-4 register loads at capture event time. 1h (R/W) = Enable CAP1-4 register loads at capture event time.
7	CTRRST4	R/W	0h	Counter Reset on Capture Event 4 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 4 (absolute time stamp operation) 1h (R/W) = Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)
6	CAP4POL	R/W	0h	Capture Event 4 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 4 triggered on a rising edge (RE) 1h (R/W) = Capture Event 4 triggered on a falling edge (FE)
5	CTRRST3	R/W	0h	Counter Reset on Capture Event 3 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 3 (absolute time stamp) 1h (R/W) = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)

**Table 14-12. ECCTL1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	CAP3POL	R/W	0h	Capture Event 3 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 3 triggered on a rising edge (RE) 1h (R/W) = Capture Event 3 triggered on a falling edge (FE)
3	CTRRST2	R/W	0h	Counter Reset on Capture Event 2 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 2 (absolute time stamp) 1h (R/W) = Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)
2	CAP2POL	R/W	0h	Capture Event 2 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 2 triggered on a rising edge (RE) 1h (R/W) = Capture Event 2 triggered on a falling edge (FE)
1	CTRRST1	R/W	0h	Counter Reset on Capture Event 1 Reset type: SYSRSn 0h (R/W) = Do not reset counter on Capture Event 1 (absolute time stamp) 1h (R/W) = Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)
0	CAP1POL	R/W	0h	Capture Event 1 Polarity select Reset type: SYSRSn 0h (R/W) = Capture Event 1 triggered on a rising edge (RE) 1h (R/W) = Capture Event 1 triggered on a falling edge (FE)

### 14.9.2.9 ECCTL2 Register (Offset = 15h) [Reset = 0006h]

ECCTL2 is shown in [Figure 14-25](#) and described in [Table 14-13](#).

Return to the [Summary Table](#).

Capture Control Register 2

**Figure 14-25. ECCTL2 Register**

15	14	13	12	11	10	9	8
MODCNRSTS		RESERVED		CTRFILTRESE T	APWMPOL	CAP_APWM	SWSYNC
R-0h		R/W-0h		R-0/W1C-0h	R/W-0h	R/W-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
SYNCO_SEL		SYNCl_EN	TSCTRSTOP	REARM	STOP_WRAP		CONT_ONESH T
R/W-0h		R/W-0h	R/W-0h	R-0/W1S-0h	R/W-3h		R/W-0h

**Table 14-13. ECCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	MODCNRSTS	R	0h	This bit field reads current status on modulo counter 00b (R) = CAP1 register gets loaded on next capture event. 01b (R) = CAP2 register gets loaded on next capture event. 10b (R) = CAP3 register gets loaded on next capture event. 11b (R) = CAP4 register gets loaded on next capture event. Reset type: CPU1.SYSRSn
13-12	RESERVED	R/W	0h	Reserved
11	CTRFILTRESET	R-0/W1C	0h	Reset Bit 0h (R) = No effect 1h (W) = Resets event filter, counter, modulo counter and CEVT[1,2,3,4] and CNTOVF , HRERROR flags Note: This provides an ability start capture module from known state in case spurious inputs are captured while ECAP is configured. Reset type: CPU1.SYSRSn
10	APWMPOL	R/W	0h	APWM output polarity select. This is applicable only in APWM operating mode. Reset type: SYSRSn 0h (R/W) = Output is active high (Compare value defines high time) 1h (R/W) = Output is active low (Compare value defines low time)
9	CAP_APWM	R/W	0h	CAP/APWM operating mode select Reset type: SYSRSn 0h (R/W) = ECAP module operates in capture mode. This mode forces the following configuration: - Inhibits TSCTR resets via CTR = PRD event - Inhibits shadow loads on CAP1 and 2 registers - Permits user to enable CAP1-4 register load - CAPx/APWMx pin operates as a capture input 1h (R/W) = ECAP module operates in APWM mode. This mode forces the following configuration: - Resets TSCTR on CTR = PRD event (period boundary) - Permits shadow loading on CAP1 and 2 registers - Disables loading of time-stamps into CAP1-4 registers - CAPx/APWMx pin operates as a APWM output

**Table 14-13. ECCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8	SWSYNC	R-0/W1S	0h	Software-forced Counter (TSCTR) Synchronizer. This provides the user a method to generate a synchronization pulse through software. In APWM mode, the synchronization pulse can also be sourced from the CTR = PRD event. Reset type: SYSRSn 0h (R/W) = Writing a zero has no effect. Reading always returns a zero 1h (R/W) = Writing a one forces a TSCTR shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing a 1, this bit returns to a zero.
7-6	SYNCO_SEL	R/W	0h	Sync-Out Select Reset type: SYSRSn 0h (R/W) = sync out signal is SWSYNC 1h (R/W) = Select CTR = PRD event to be the sync-out signal. Note: Selection CTR = PRD is meaningful only in APWM mode 2h (R/W) = Disable sync out signal 3h (R/W) = Disable sync out signal
5	SYNCI_EN	R/W	0h	Counter (TSCTR) Sync-In select mode Reset type: SYSRSn 0h (R/W) = Disable sync-in option 1h (R/W) = Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCI signal or a S/W force event.
4	TSCTRSTOP	R/W	0h	Time Stamp (TSCTR) Counter Stop (freeze) Control Reset type: SYSRSn 0h (R/W) = TSCTR stopped 1h (R/W) = TSCTR free-running
3	REARM	R-0/W1S	0h	Re-Arming Control. Note: The re-arm function is valid in one shot or continuous mode Reset type: SYSRSn 0h (R/W) = Has no effect (reading always returns a 0) 1h (R/W) = Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero 2) Unfreezes the Mod4 counter 3) Enables capture register loads
2-1	STOP_WRAP	R/W	3h	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. Reset type: SYSRSn 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) Reset type: SYSRSn 0h (R/W) = Operate in continuous mode 1h (R/W) = Operate in one-Shot mode

### 14.9.2.10 ECEINT Register (Offset = 16h) [Reset = 0000h]

ECEINT is shown in [Figure 14-26](#) and described in [Table 14-14](#).

Return to the [Summary Table](#).

The interrupt enable bits (CEVT1, ...) block any of the selected events from generating an interrupt. Events will still be latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers. The proper procedure for configuring peripheral modes and interrupts is as follows:

- Disable global interrupts
- Stop eCAP counter
- Disable eCAP interrupts
- Configure peripheral registers
- Clear spurious eCAP interrupt flags
- Enable eCAP interrupts
- Start eCAP counter
- Enable global interrupts

**Figure 14-26. ECEINT Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							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-0h

**Table 14-14. ECEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	CTR_EQ_CMP	R/W	0h	Counter Equal Compare Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Compare Equal as an Interrupt source 1h (R/W) = Enable Compare Equal as an Interrupt source
6	CTR_EQ_PRD	R/W	0h	Counter Equal Period Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Period Equal as an Interrupt source 1h (R/W) = Enable Period Equal as an Interrupt source
5	CTROVF	R/W	0h	Counter Overflow Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disabled counter Overflow as an Interrupt source 1h (R/W) = Enable counter Overflow as an Interrupt source
4	CEVT4	R/W	0h	Capture Event 4 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 4 as an Interrupt source 1h (R/W) = Capture Event 4 Interrupt Enable
3	CEVT3	R/W	0h	Capture Event 3 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 3 as an Interrupt source 1h (R/W) = Enable Capture Event 3 as an Interrupt source
2	CEVT2	R/W	0h	Capture Event 2 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 2 as an Interrupt source 1h (R/W) = Enable Capture Event 2 as an Interrupt source

**Table 14-14. ECEINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	CEVT1	R/W	0h	Capture Event 1 Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disable Capture Event 1 as an Interrupt source 1h (R/W) = Enable Capture Event 1 as an Interrupt source
0	RESERVED	R	0h	Reserved

### 14.9.2.11 ECFLG Register (Offset = 17h) [Reset = 0000h]

ECFLG is shown in [Figure 14-27](#) and described in [Table 14-15](#).

Return to the [Summary Table](#).

Capture Interrupt Flag Register

**Figure 14-27. ECFLG Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							R-0h
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 14-15. ECFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R	0h	Reserved
7	CTR_CMP	R	0h	Compare Equal Compare Status Flag. This flag is active only in APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) reached the compare register value (ACMP)
6	CTR_PRD	R	0h	Counter Equal Period Status Flag. This flag is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) reached the period register value (APRD) and was reset.
5	CTROVF	R	0h	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the counter (TSCTR) has made the transition from FFFFFFFF to 00000000
4	CEVT4	R	0h	Capture Event 4 Status Flag This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the fourth event occurred at ECAPx pin
3	CEVT3	R	0h	Capture Event 3 Status Flag. This flag is active only in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the third event occurred at ECAPx pin.
2	CEVT2	R	0h	Capture Event 2 Status Flag. This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the second event occurred at ECAPx pin.
1	CEVT1	R	0h	Capture Event 1 Status Flag. This flag is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates the first event occurred at ECAPx pin.

**Table 14-15. ECFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	INT	R	0h	Global Interrupt Status Flag Reset type: SYSRSn 0h (R/W) = Indicates no event occurred 1h (R/W) = Indicates that an interrupt was generated.

### 14.9.2.12 ECCLR Register (Offset = 18h) [Reset = 0000h]

ECCLR is shown in [Figure 14-28](#) and described in [Table 14-16](#).

Return to the [Summary Table](#).

Capture Interrupt Clear Register

**Figure 14-28. ECCLR Register**

15	14	13	12	11	10	9	8
RESERVED							RESERVED
R-0h							R-0/W1C-0h
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R-0/W1C-0h							

**Table 14-16. ECCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R-0/W1C	0h	Reserved
7	CTR_CMP	R-0/W1C	0h	Counter Equal Compare Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTR=CMP flag.
6	CTR_PRD	R-0/W1C	0h	Counter Equal Period Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTR=PRD flag.
5	CTROVF	R-0/W1C	0h	Counter Overflow Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CTROVF flag.
4	CEVT4	R-0/W1C	0h	Capture Event 4 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT4 flag.
3	CEVT3	R-0/W1C	0h	Capture Event 3 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT3 flag.
2	CEVT2	R-0/W1C	0h	Capture Event 2 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT2 flag.
1	CEVT1	R-0/W1C	0h	Capture Event 1 Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the CEVT1 flag.
0	INT	R-0/W1C	0h	ECAP Global Interrupt Status Clear Reset type: SYSRSn 0h (R/W) = Writing a 0 has no effect. Always reads back a 0 1h (R/W) = Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1

### 14.9.2.13 ECFRC Register (Offset = 19h) [Reset = 0000h]

ECFRC is shown in [Figure 14-29](#) and described in [Table 14-17](#).

Return to the [Summary Table](#).

Capture Interrupt Force Register

**Figure 14-29. ECFRC Register**

15		14		13		12		11		10		9		8	
RESERVED													RESERVED		
R-0h													R-0/W1S-0h		
7		6		5		4		3		2		1		0	
CTR_CMP		CTR_PRD		CTROVF		CEVT4		CEVT3		CEVT2		CEVT1		RESERVED	
R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0/W1S-0h		R-0h	

**Table 14-17. ECFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reserved
8	RESERVED	R-0/W1S	0h	Reserved
7	CTR_CMP	R-0/W1S	0h	Force Counter Equal Compare Interrupt. This event is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR_CMP flag.
6	CTR_PRD	R-0/W1S	0h	Force Counter Equal Period Interrupt. This event is only active in APWM mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CTR_PRD flag.
5	CTROVF	R-0/W1S	0h	Force Counter Overflow Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 to this bit sets the CTROVF flag.
4	CEVT4	R-0/W1S	0h	Force Capture Event 4. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT4 flag.
3	CEVT3	R-0/W1S	0h	Force Capture Event 3. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT3 flag.
2	CEVT2	R-0/W1S	0h	Force Capture Event 2. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Writing a 1 sets the CEVT2 flag.
1	CEVT1	R-0/W1S	0h	Force Capture Event 1. This event is only active in CAP mode. Reset type: SYSRSn 0h (R/W) = No effect. Always reads back a 0. 1h (R/W) = Sets the CEVT1 flag.
0	RESERVED	R	0h	Reserved

### 14.9.2.14 ECAPSYNCINSEL Register (Offset = 1Eh) [Reset = 0000001h]

ECAPSYNCINSEL is shown in [Figure 14-30](#) and described in [Table 14-18](#).

Return to the [Summary Table](#).

SYNC source select register

**Figure 14-30. ECAPSYNCINSEL Register**

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

**Table 14-18. ECAPSYNCINSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-0	SEL	R/W	1h	These bits determines the source of SYNCIN signal. Reset type: SYSRSn 0h (R/W) = Disable Syncin to eCAP 1h (R/W) = EPWM1SYNCOULT 2h (R/W) = EPWM2SYNCOULT 3h (R/W) = EPWM3SYNCOULT 4h (R/W) = EPWM4SYNCOULT 5h (R/W) = EPWM5SYNCOULT 6h (R/W) = EPWM6SYNCOULT 7h (R/W) = EPWM7SYNCOULT 8h (R/W) = RSVD 9h (R/W) = RSVD Ah (R/W) = RSVD Bh (R/W) = RSVD Ch (R/W) = RSVD Dh (R/W) = RSVD Eh (R/W) = RSVD Fh (R/W) = RSVD 10h (R/W) = RSVD 11h (R/W) = ECAP1SYNCOULT 12h (R/W) = ECAP2SYNCOULT 13h (R/W) = RSVD 14h (R/W) = RSVD 15h (R/W) = RSVD 16h (R/W) = RSVD 17h (R/W) = RSVD 18h (R/W) = INPUTXBAROUT5 19h (R/W) = INPUTXBAROUT6 1Ah (R/W) = RSVD 1Bh (R/W) = RSVD 1Ch (R/W) = RSVD 1Dh (R/W) = RSVD 1Eh (R/W) = RSVD 1Fh (R/W) = RSVD

### 14.9.3 ECAP Registers to Driverlib Functions

**Table 14-19. ECAP Registers to Driverlib Functions**

File	Driverlib Function
<b>TSCTR</b>	
ecap.h	ECAP_getTimeBaseCounter
<b>CTRPHS</b>	
ecap.h	ECAP_setPhaseShiftCount
<b>CAP1</b>	
ecap.h	ECAP_setAPWMPeriod

**Table 14-19. ECAP Registers to Driverlib Functions (continued)**

File	Driverlib Function
ecap.h	ECAP_getEventTimeStamp
<b>CAP2</b>	
ecap.h	ECAP_setAPWMCompare
ecap.h	ECAP_getEventTimeStamp
<b>CAP3</b>	
ecap.h	ECAP_setAPWMShadowPeriod
ecap.h	ECAP_getEventTimeStamp
<b>CAP4</b>	
ecap.h	ECAP_setAPWMShadowCompare
ecap.h	ECAP_getEventTimeStamp
<b>ECCTL0</b>	
ecap.h	ECAP_selectECAPInput
<b>ECCTL1</b>	
ecap.c	ECAP_setEmulationMode
ecap.h	ECAP_setEventPrescaler
ecap.h	ECAP_setEventPolarity
ecap.h	ECAP_enableCounterResetOnEvent
ecap.h	ECAP_disableCounterResetOnEvent
ecap.h	ECAP_enableTimeStampCapture
ecap.h	ECAP_disableTimeStampCapture
<b>ECCTL2</b>	
ecap.h	ECAP_setCaptureMode
ecap.h	ECAP_reArm
ecap.h	ECAP_enableCaptureMode
ecap.h	ECAP_enableAPWMMode
ecap.h	ECAP_enableLoadCounter
ecap.h	ECAP_disableLoadCounter
ecap.h	ECAP_loadCounter
ecap.h	ECAP_setSyncOutMode
ecap.h	ECAP_stopCounter
ecap.h	ECAP_startCounter
ecap.h	ECAP_setAPWMPolarity
ecap.h	ECAP_resetCounters
ecap.h	ECAP_getModuloCounterStatus
<b>ECEINT</b>	
ecap.h	ECAP_enableInterrupt
ecap.h	ECAP_disableInterrupt
<b>ECFLG</b>	
ecap.h	ECAP_getInterruptSource
ecap.h	ECAP_getGlobalInterruptStatus
<b>ECCLR</b>	
ecap.h	ECAP_clearInterrupt
ecap.h	ECAP_clearGlobalInterrupt
<b>ECFRC</b>	
ecap.h	ECAP_forceInterrupt

**Table 14-19. ECAP Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>SYNCINSEL</b>	
ecap.h	ECAP_setSynclnPulseSource

# Chapter 15

## Enhanced Quadrature Encoder Pulse (eQEP)

---



The enhanced Quadrature Encoder Pulse (eQEP) module described here is a Type 2 eQEP. See the [C2000 Real-Time Control Peripheral Reference Guide](#) 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.

<b>15.1 Introduction</b> .....	<b>1558</b>
<b>15.2 Configuring Device Pins</b> .....	<b>1560</b>
<b>15.3 Description</b> .....	<b>1561</b>
<b>15.4 Quadrature Decoder Unit (QDU)</b> .....	<b>1566</b>
<b>15.5 Position Counter and Control Unit (PCCU)</b> .....	<b>1569</b>
<b>15.6 eQEP Edge Capture Unit</b> .....	<b>1577</b>
<b>15.7 eQEP Watchdog</b> .....	<b>1581</b>
<b>15.8 eQEP Unit Timer Base</b> .....	<b>1581</b>
<b>15.9 QMA Module</b> .....	<b>1582</b>
<b>15.10 eQEP Interrupt Structure</b> .....	<b>1585</b>
<b>15.11 Software</b> .....	<b>1586</b>
<b>15.12 eQEP Registers</b> .....	<b>1589</b>

### 15.1 Introduction

An incremental encoder disk is patterned with a track of slots along the periphery, as shown in Figure 15-1. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark and light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

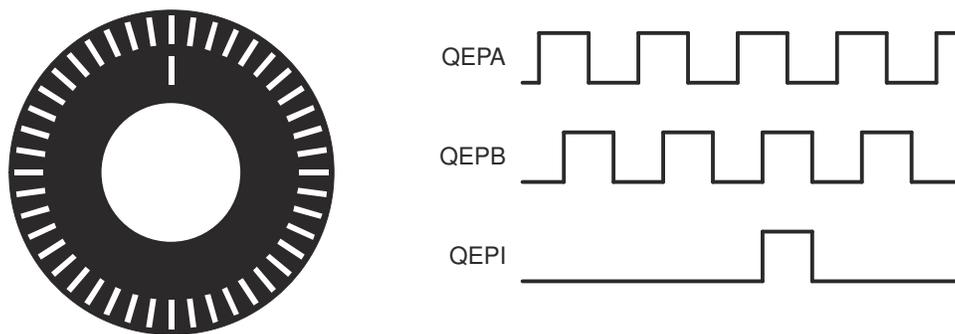


Figure 15-1. Optical Encoder Disk

To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is detected with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and conversely, as shown in Figure 15-2.

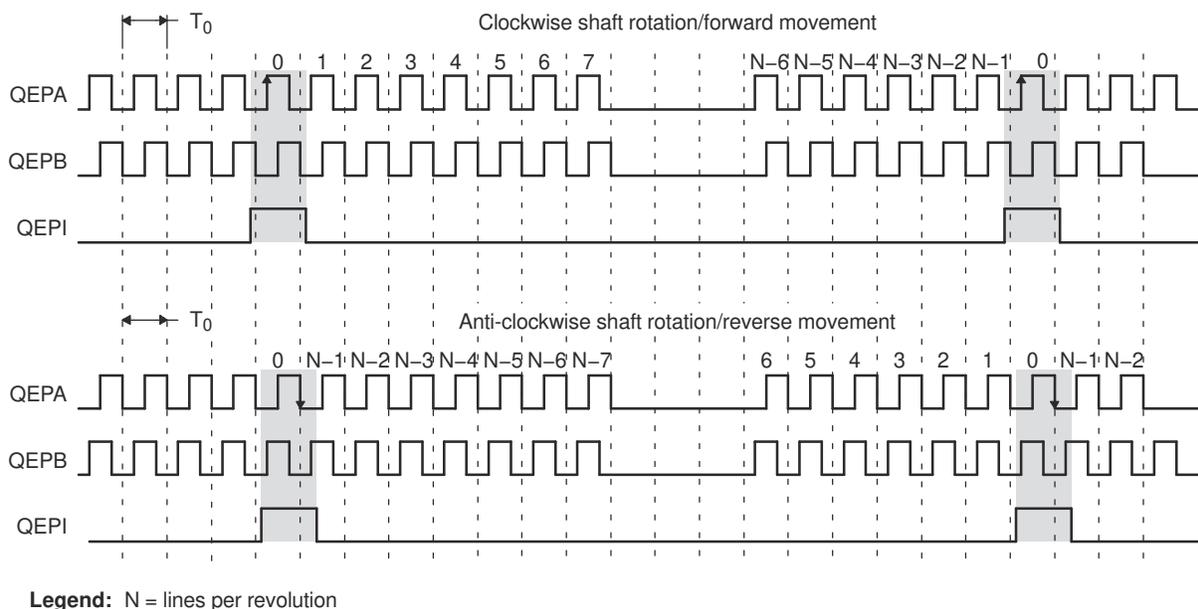
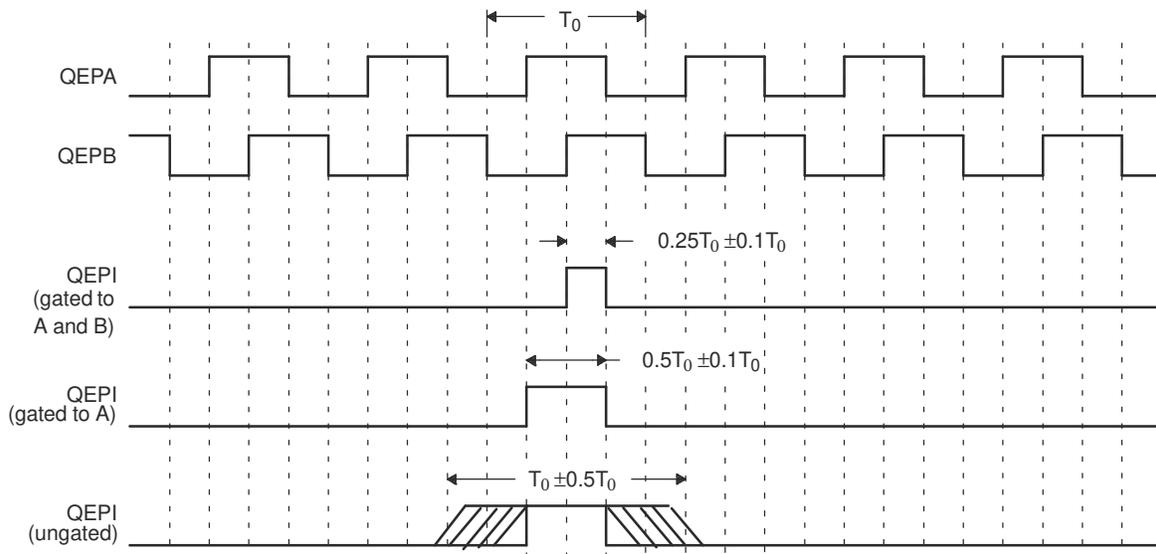


Figure 15-2. QEP Encoder Output Signal for Forward/Reverse Movement

The encoder wheel typically makes one revolution for every revolution of the motor, or the wheel can 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 166.6kHz, 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 15-3. 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 15-3. Index Pulse Example**

Some typical applications of shaft encoders include robotics and computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

**General Issues:** Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity can be written as:

$$v(k) \approx \frac{x(k) - x(k - 1)}{T} = \frac{\Delta X}{T} \quad (11)$$

$$v(k) \approx \frac{X}{t(k) - t(k - 1)} = \frac{X}{\Delta T} \quad (12)$$

where:

- $v(k)$  = Velocity at time instant  $k$
- $x(k)$  = Position at time instant  $k$
- $x(k-1)$  = Position at time instant  $k-1$
- $T$  = Fixed unit time or inverse of velocity calculation rate
- $\Delta X$  = Incremental position movement in unit time
- $t(k)$  = Time instant " $k$ "
- $t(k-1)$  = Time instant " $k-1$ "
- $X$  = Fixed unit position
- $\Delta T$  = Incremental time elapsed for unit position movement

[Equation 11](#) is the conventional approach to velocity estimation and requires a time base to provide a unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity  $[x(k) - x(k-1)]$  is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant  $1/T$  (where  $T$  is the constant time between unit time events and is known in advance).

Estimation based on [Equation 11](#) 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 400Hz. 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 12rpm when sampled at 400Hz. While this resolution can be satisfactory at moderate or high speeds, for example 1% error at 1200rpm, this resolution clearly proves inadequate at low speeds. In fact, at speeds below 12rpm, the speed estimate is erroneously zero much of the time.

At low speed, [Equation 12](#) 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 12](#) 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 11](#). A combination of relatively large motor speeds and high sensor resolution makes the time interval  $\Delta T$  small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use [Equation 12](#) at low speed and have the DSP software switch over to [Equation 11](#) when the motor speed rises above some specified threshold.

### 15.1.1 EQEP Related Collateral

#### Foundational Materials

- [C2000 Academy - EQEP](#)
- [Interfacing with Quadrature Encoders](#) (Video)
- [Real-Time Control Reference Guide](#)
  - Refer to the Encoders section

#### Getting Started Materials

- [C2000™ Position Manager PTO API Reference Guide Application Report](#)

#### Expert Materials

- [CW/CCW Support on the C2000 eQEP Module Application Report](#)

## 15.2 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 using the GPxQSELn registers for synchronous input mode (with or without qualification). The asynchronous mode cannot be used for eQEP input pins. The internal pullups can be configured in the GPyPUD register.

See the *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux and settings.

## 15.3 Description

This section provides the eQEP inputs, memory map, and functional description.

### 15.3.1 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 can enable the synchronous GPIO input feature on any eQEP-enabled GPIO pins (see the *General-Purpose Input/Output (GPIO)* chapter for more details).

- **QEPA/XCLK and QEPB/XDIR:** These two pins can be used in quadrature-clock mode or direction-count mode.
  - Quadrature-clock Mode: The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase. This phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and conversely. 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.

Input signals to the eQEP (QEPA, QEPB, QEPI and QEPS) can come from multiple sources; that is, device pin, CMPSSx, or PWMXBARx. One typical use case is if SinCos transducers are used in the motor control system to estimate the position of motor shaft and Index signal is coming from traditional rotary encoder, source of the eQEP signals (QEPA, QEPB and QEPI) can be configured as output of CMPSSx that decodes the Sin, Cos, and Index signals. [Figure 15-4](#) illustrates the use case.

Selection of the source of Input signals (QEPA, QEPB, and QEPI) is user-configurable through the QEPSRCSEL register as shown in [Table 15-1](#).

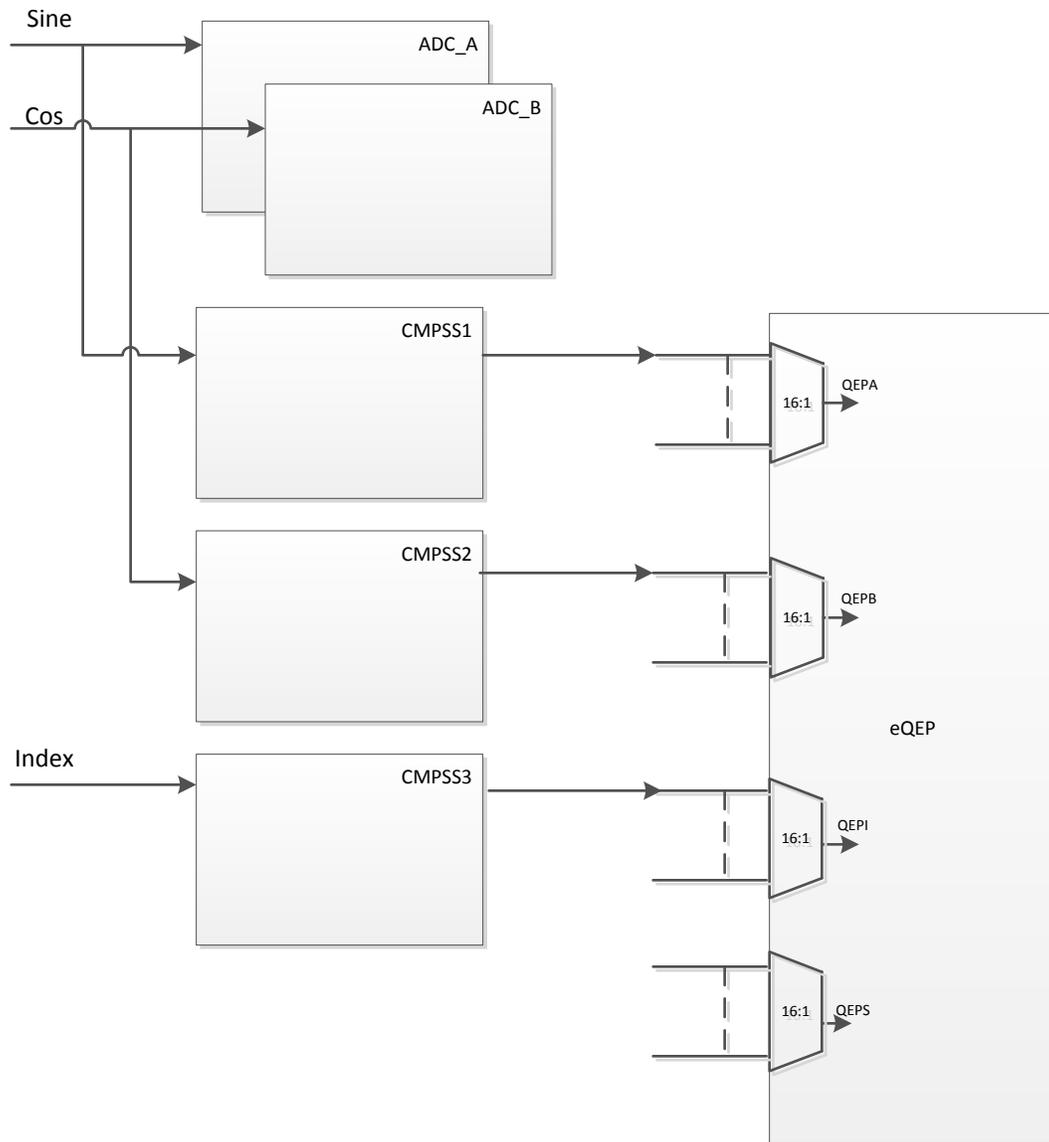


Figure 15-4. Using eQEP to Decode Signals from SinCos Transducer

**Table 15-1. eQEP Input Source Select Table**

QEPASEL, QEPBSEL, QEPISEL	Input Signal
0	INPUTXBAR
1	CMPSS1_CTRIPH
2	CMPSS2_CTRIPH
3	CMPSS3_CTRIPH
4	CMPSS4_CTRIPH
5-7	Reserved
8	ZERO
9	EPWMXBAR1
10	EPWMXBAR2
11	EPWMXBAR3
12	EPWMXBAR4
13	EPWMXBAR5
14	EPWMXBAR6
15	EPWMXBAR7

---

**Note**

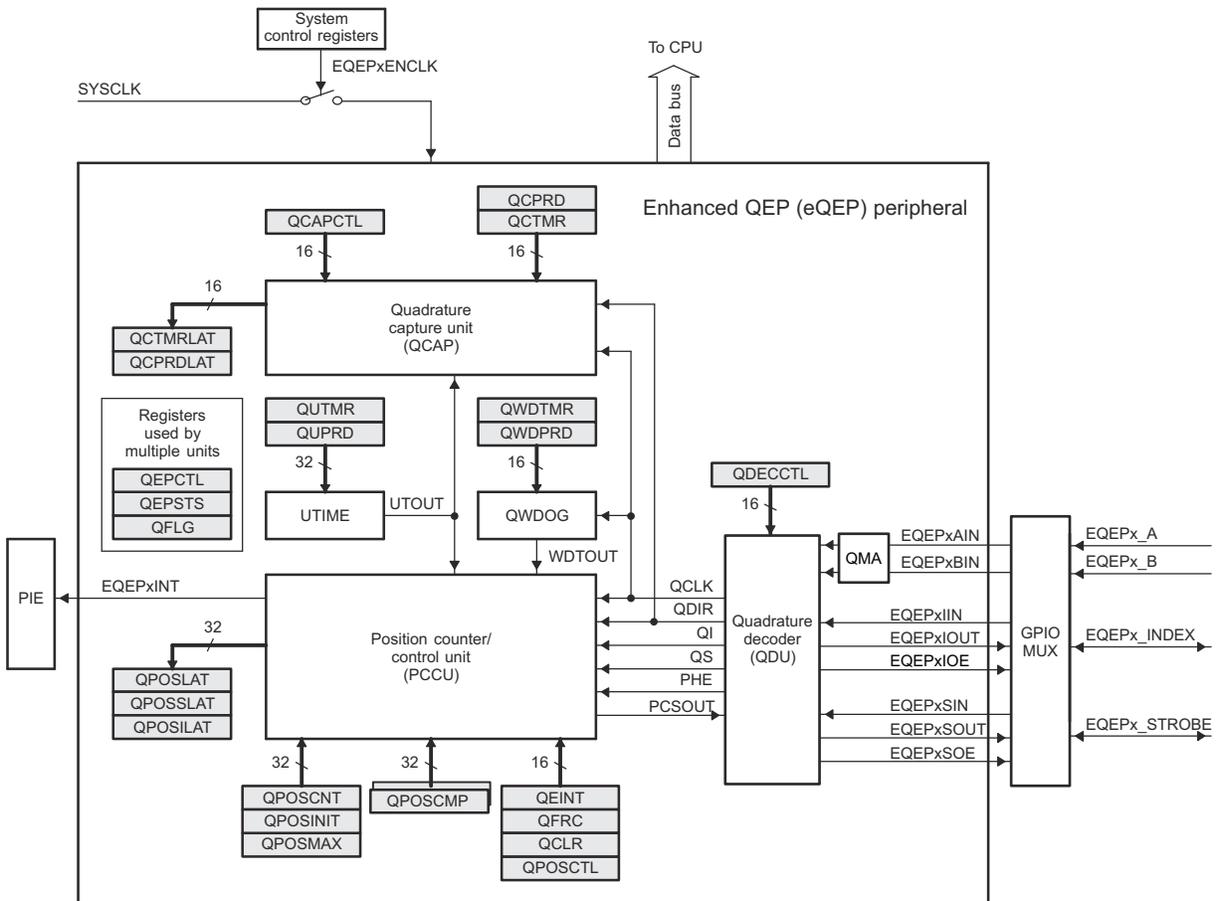
Configuration of QEPSRCSEL register to select the source of QEPA, QEPB, and QEPI signals can lead to unexpected transition on these signals, which can cause an undesirable outcome if eQEP is already running. Make sure that the eQEP is disabled before configuring the QEPSRCSEL register for input signals.

---

### 15.3.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 15-5):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)
- Quadrature Mode Adapter (QMA)



Copyright © 2017, Texas Instruments Incorporated

Figure 15-5. Functional Block Diagram of the eQEP Peripheral

### 15.3.3 eQEP Memory Map

Table 15-2 lists the registers with the memory locations, sizes, and reset values.

**Table 15-2. EQEP Memory Map**

Name	Offset	Size(x16)/ #shadow	Reset	Register Description
QPOSCNT	0x00	2/0	0x0000 0000	eQEP Position Counter
QPOSINIT	0x02	2/0	0x0000 0000	eQEP Initialization Position Count
QPOSMAX	0x04	2/0	0x0000 0000	eQEP Maximum Position Count
QPOSCMP	0x06	2/1	0x0000 0000	eQEP Position-compare
QPOSILAT	0x08	2/0	0x0000 0000	eQEP Index Position Latch
QPOSSLAT	0x0A	2/0	0x0000 0000	eQEP Strobe Position Latch
QPOSLAT	0x0C	2/0	0x0000 0000	eQEP Position Latch
QUTMR	0x0E	2/0	0x0000 0000	eQEP Unit Timer
QUPRD	0x10	2/0	0x0000 0000	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 0x2F	15/0		
REV	0x30	2/0	0x0000	eQEP Revision Number
QEPSTROBESEL	0x32	2/0	0x0000	eQEP Strobe select register
QMACTRL	0x34	2/0	0x0000	eQEP QMA Control register
QEPSRCSEL	0x36	2/0	0x0000	eQEP Source Select Register
Reserved	0x38 to 0x3F	8/0		



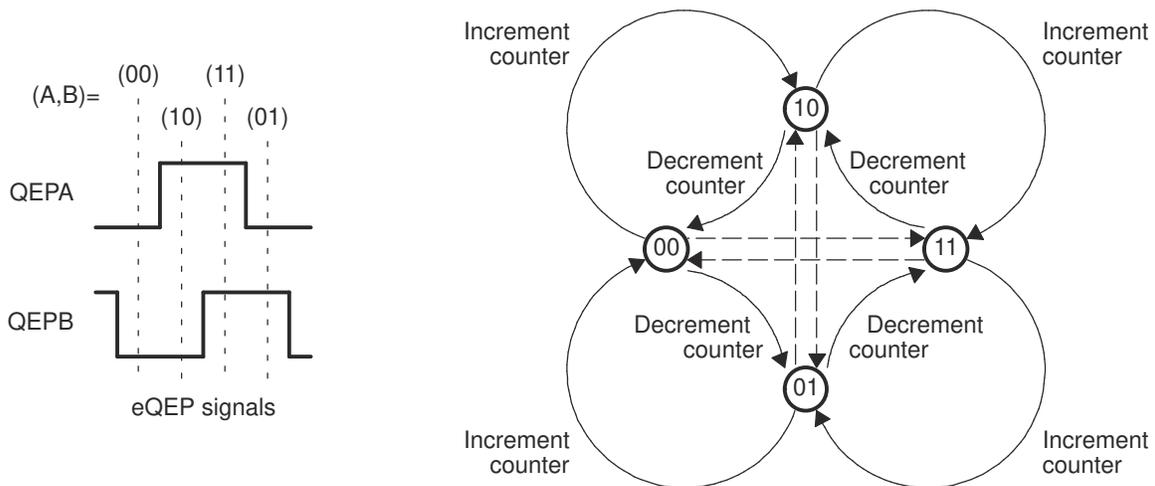
### 15.4.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

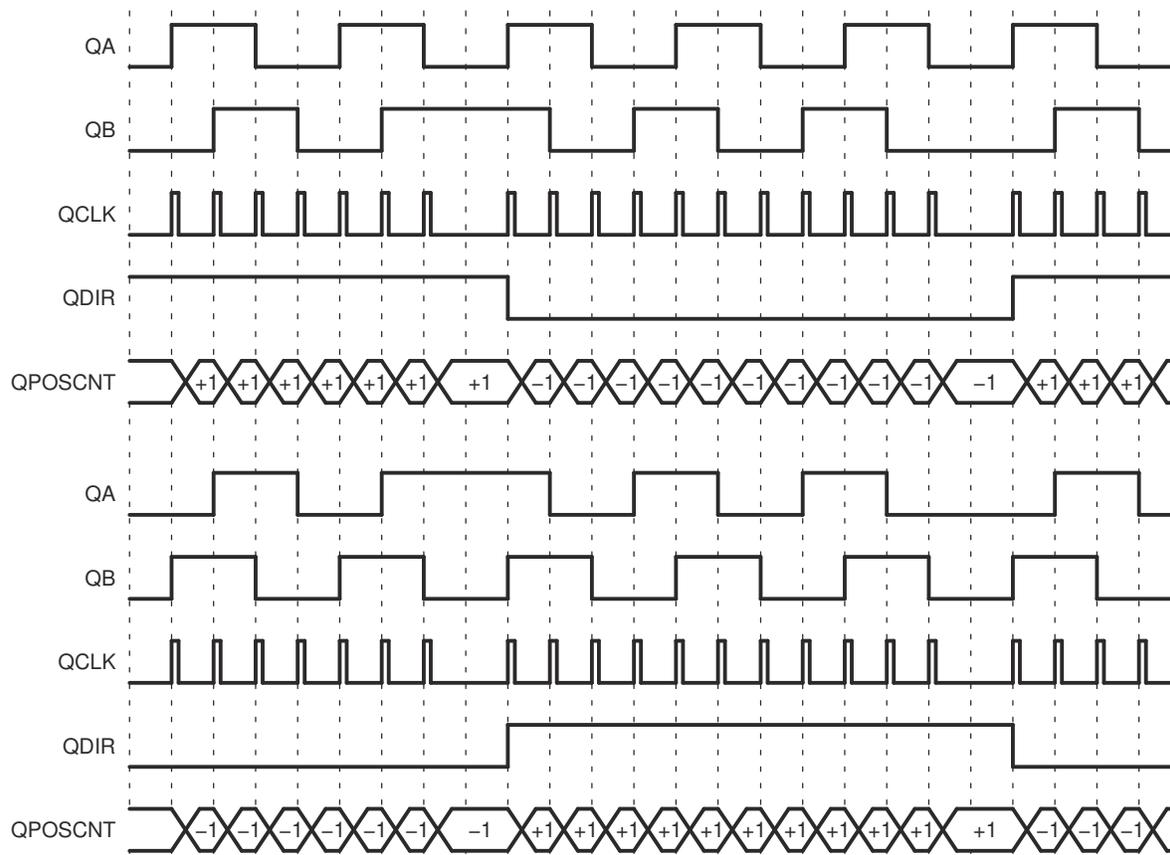
**Direction Decoding** The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in the QEPSTS[QDF] bit. Table 15-3 and Figure 15-7 show the direction decoding logic in truth table and 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 15-8 shows the direction decoding and clock generation from the eQEP input signals.

**Table 15-3. 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	Decrement
	QA↓	UP	Increment
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Decrement
	QA↑	UP	Increment
	QB↑	TOGGLE	Increment or Decrement



**Figure 15-7. Quadrature Decoder State Machine**



**Figure 15-8. Quadrature-clock and Direction Decoding**

**Phase Error Flag** In normal operating conditions, quadrature inputs QEPA and QEPB is 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register and the QPOSCNT value can be incorrect and offset by multiples of 1 or 3. That is, when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in [Figure 15-7](#) 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 15-8](#).

**Reverse Count** In normal quadrature count operation, QEPA input is applied to the QA input of the quadrature decoder and the QEPB input is applied to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This swaps the input to the quadrature decoder; thereby, reversing the counting direction.

#### 15.4.1.2 Direction-Count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. The QEPA input provides the clock for the position counter and the QEPB input has 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.

### 15.4.1.3 Up-Count Mode

The counter direction signal is hard-wired for up-count and the position counter is used to measure the frequency of the QEPA input. Clearing the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input; thereby, increasing the measurement resolution by a factor of 2x. In up-count mode, we recommend that the application not configure QEPB as a GPIO mux option, or make sure that a signal edge is not generated on the QEPB input.

### 15.4.1.4 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. Clearing the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by a factor of 2x. In down-count mode, the application must not configure QEPB as a GPIO mux option or make sure that a signal edge is not generated on the QEPB input.

### 15.4.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting the QDECCTL[QIP] bit inverts the index input.

### 15.4.3 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.

## 15.5 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.

### 15.5.1 Position Counter Operating Modes

Position-counter data can be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position-counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then the position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse, and the position counter provides a rotor angle with respect to the index pulse position.

The position counter can be configured to operate in following four modes

- Position-Counter Reset on Index Event
- Position-Counter Reset on Maximum Position
- Position-Counter Reset on the first Index Event
- Position-Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, the position counter is reset to 0 on overflow and to the QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after the QPOSMAX value. Underflow occurs when the position counter counts down after 0. The Interrupt flag is set to indicate overflow/underflow in QFLG register.

### 15.5.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM]=00)

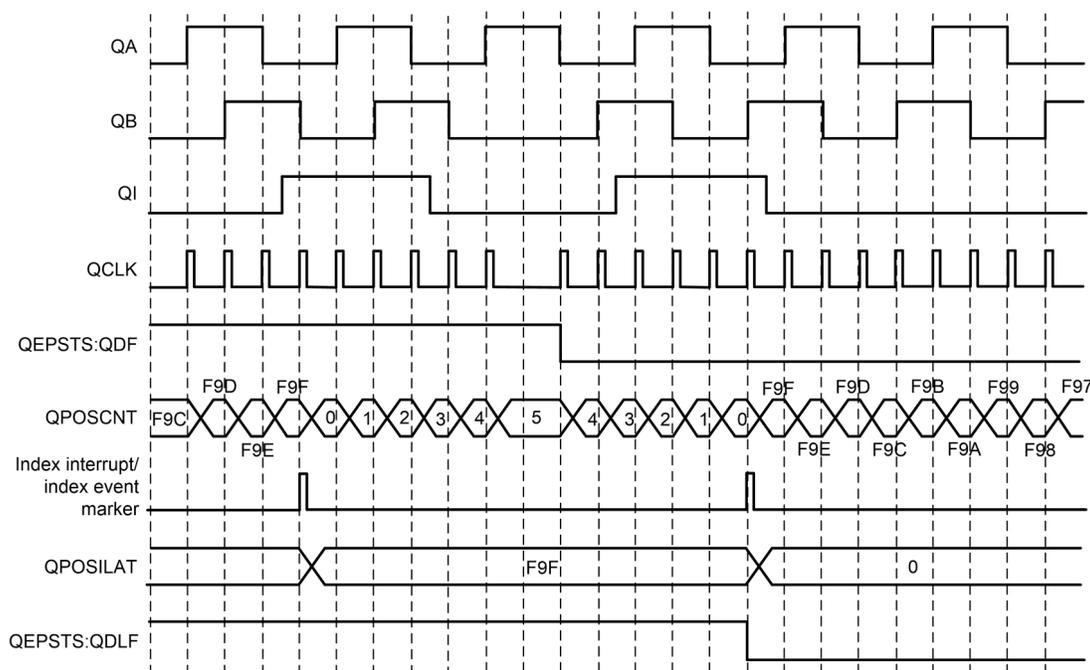
If the index event occurs during the forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOS MAX register on the next eQEP clock.

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 QEPSTS registers, the eQEP peripheral 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 15-9](#).

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 QPOS MAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) is set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] must be configured to 00 or 11 when pcr=0 and the position counter error flag/interrupt flag are generated only in index event reset mode. The position counter value is latched into the IPOS LAT register on every index marker.



**Figure 15-9. Position Counter Reset by Index Pulse for 1000-Line Encoder (QPOS MAX = 3999 or 0xF9F)**

#### Note

In case of a boundary condition where the time period between the Index Event and the previous QCLK edge is less than SYSCLK period, then QPOSCNT gets reset to zero or QPOS MAX in the same SYSCLK cycle and does not wait for the next QCLK edge to occur.

### 15.5.1.2 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 15-10 shows the position-counter reset operation in this mode.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

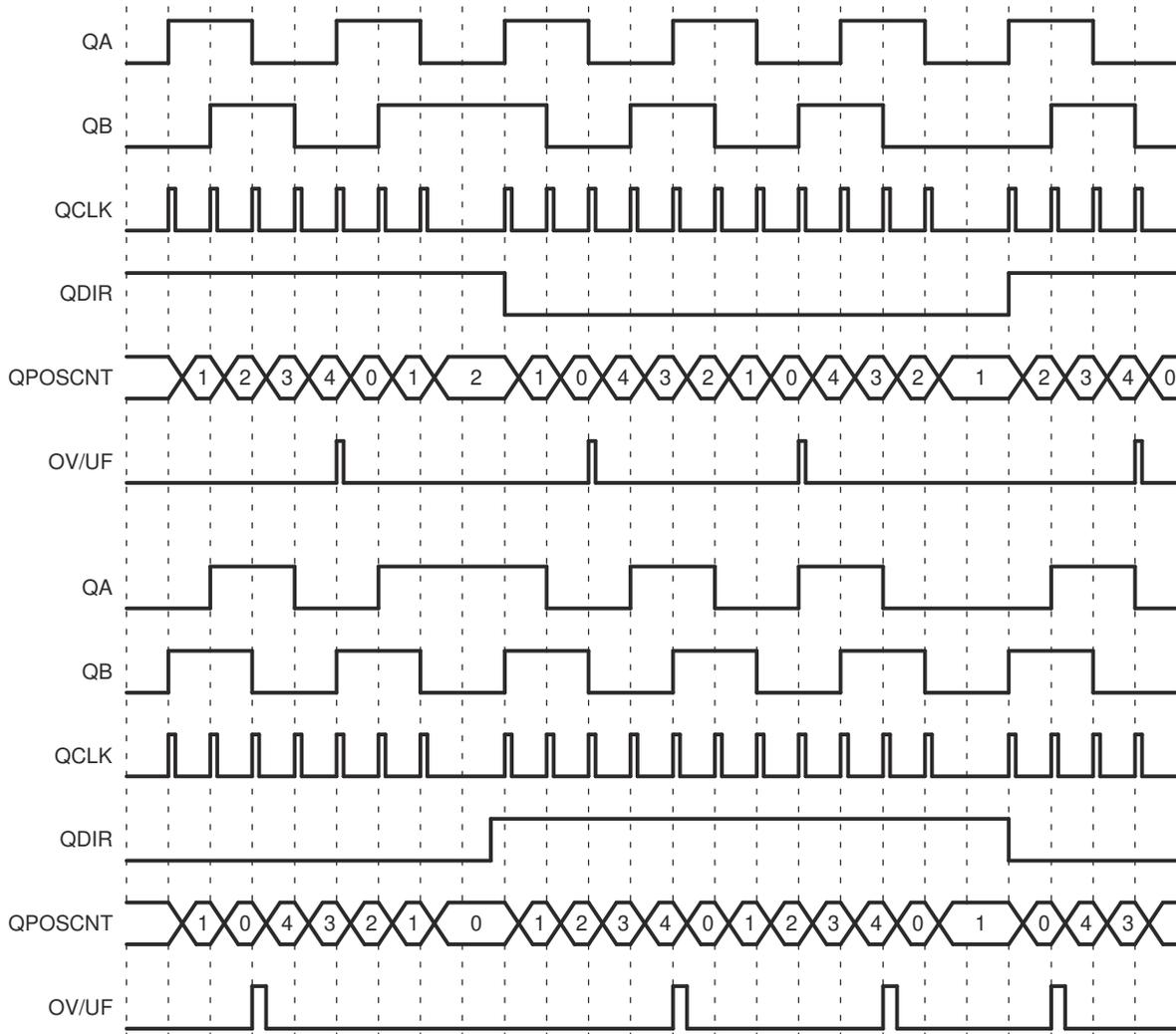


Figure 15-10. Position Counter Underflow/Overflow (QPOSMAX = 4)

### 15.5.1.3 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, the position-counter value is reset based on the maximum position as described in Section 15.5.1.2.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

#### 15.5.1.4 Position Counter Reset on Unit Time-out Event (QEPCTL[PCRM] = 11)

In this mode, QPOSCNT is set to 0 or QPOMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event. This is useful for frequency measurement.

#### 15.5.2 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.

##### 15.5.2.1 Index Event Latch

In some applications, it is not desirable to reset the position counter on every index event and instead it can be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL]=01)
- Latch on Falling edge (QEPCTL[IEL]=10)
- Latch on Index Event Marker (QEPCTL[IEL]=11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register. The index event latch configuration bits (QEPCTL[IEL]) are ignored when QEPCTL[PCRM] = 00.

<b>Latch on Rising Edge (QEPCTL[IEL]=01)</b>	The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.
<b>Latch on Falling Edge (QEPCTL[IEL] = 10)</b>	The position-counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.
<b>Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11)</b>	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 the direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. The eQEP peripheral also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 15-11 shows the position counter latch using an index event marker.

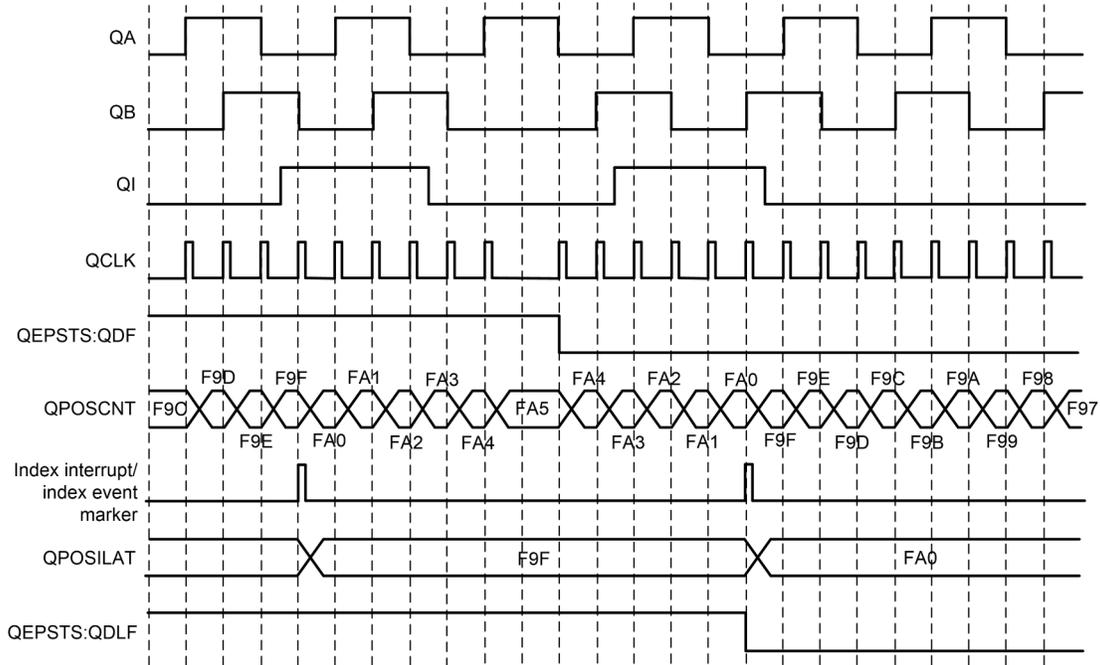


Figure 15-11. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)

### 15.5.2.2 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 15-12.

The strobe event latch interrupt flag (QLFG[SEL]) is set when the position counter is latched to the QPOSSLAT register.

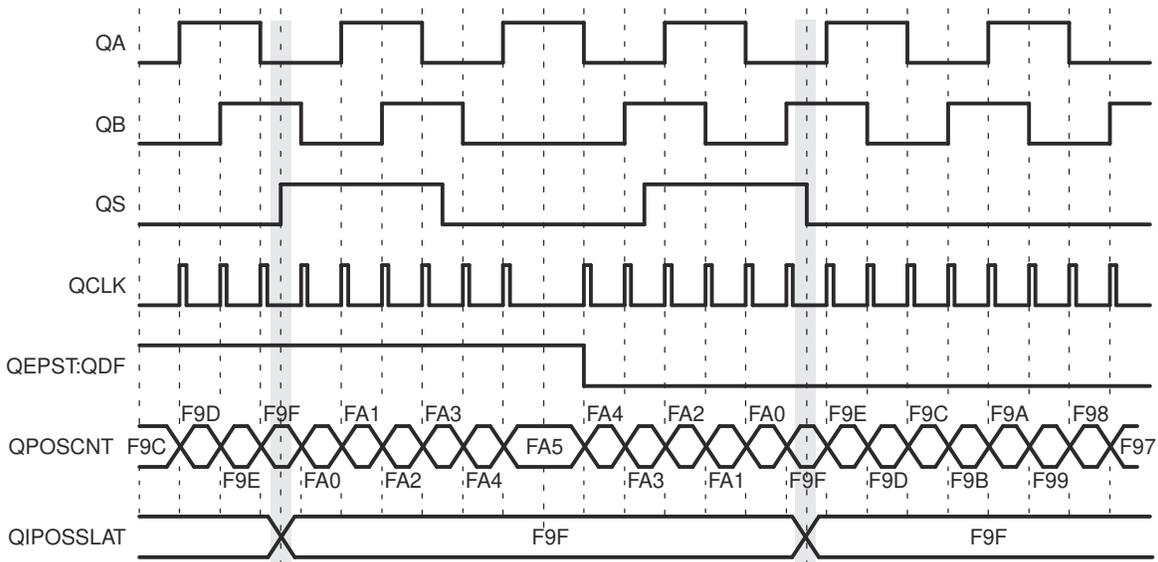
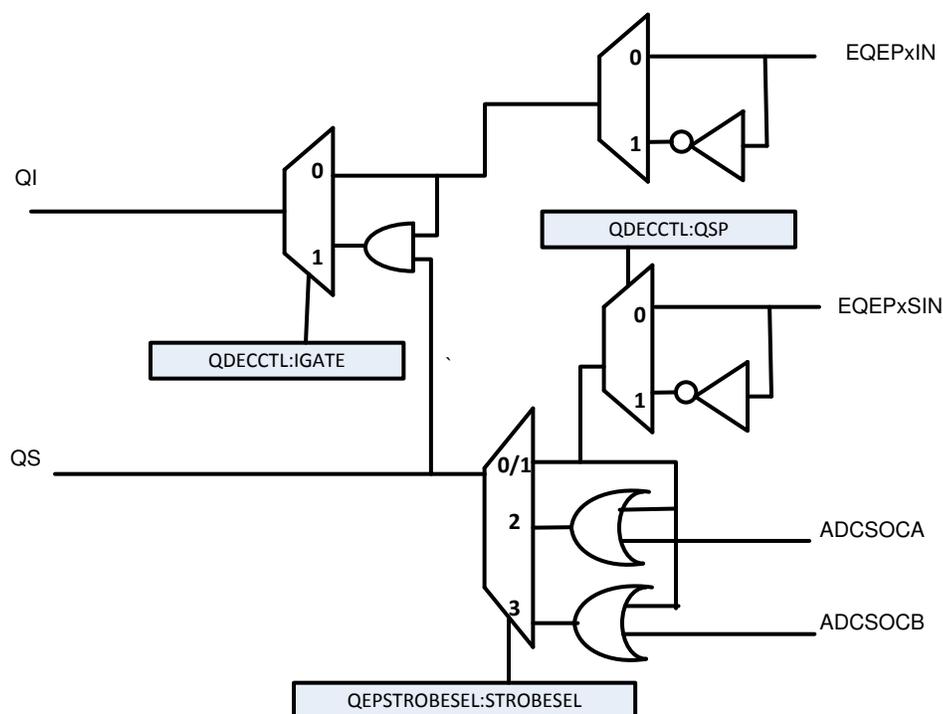


Figure 15-12. Strobe Event Latch (QEPCTL[SEL] = 1)

There is an added feature on Type 2.0 eQEP where position-counter value can also be latched on ADCSOCA and ADCSOCB events by configuring the register QEPSTROBESEL.STROBESEL as shown in Figure 15-13. To use the ADCSOCA/B events for the QS signal, configuration of the QEPSRCSEL.QEPSSEL to be non-zero is needed..



**Figure 15-13. Latching Position Counter on ADCSOCA/ADCSOCB Event**

### 15.5.3 Position Counter Initialization

The position counter can be initialized using the 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 is 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 the bit again, the position counter is re-initialized.

### 15.5.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and interrupt on a position-compare match. Figure 15-14 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.

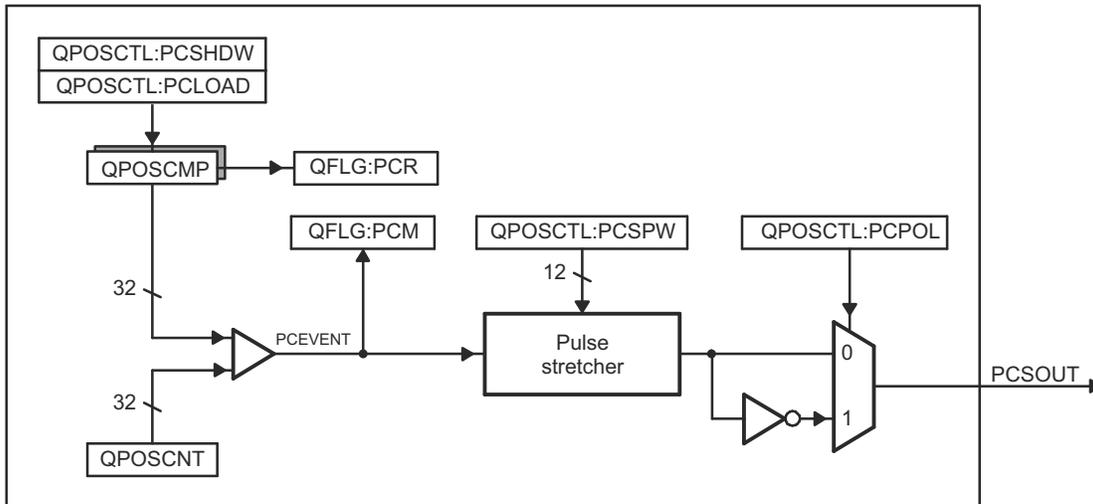


Figure 15-14. eQEP Position-compare Unit

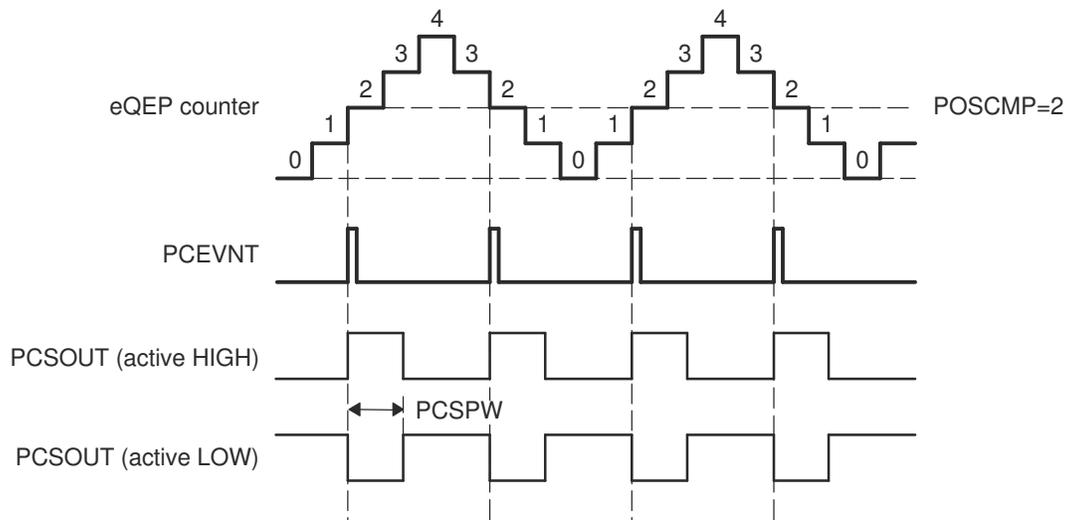
In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events, and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

- 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.

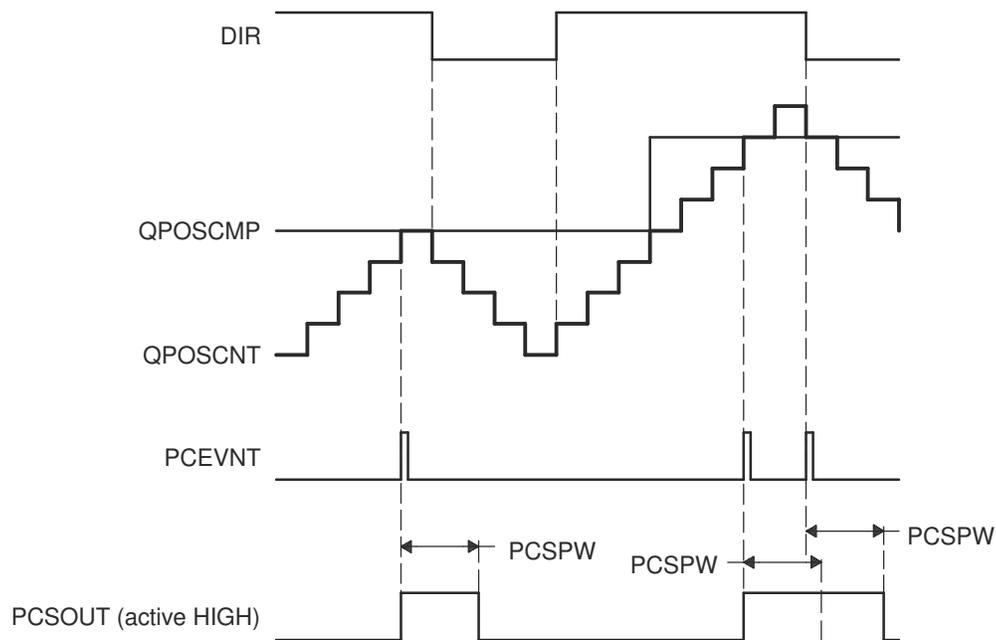
For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 15-15).

See the register section for the layout of the eQEP Position-Compare Control Register (QPOSCTL) and description of the QPOSCTL bit fields.



**Figure 15-15. 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 15-16](#).



**Figure 15-16. eQEP Position-compare Sync Output Pulse Stretcher**

## 15.6 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 15-17](#). This feature is typically used for low-speed measurement using the following formula:

$$v(k) = \frac{X}{t(k) - t(k - 1)} = \frac{X}{\Delta T} \quad (13)$$

where:

- $X$  = Unit position is defined by integer multiple of quadrature edges (see [Figure 15-18](#))
- $\Delta T$  = Elapsed time between unit position events
- $v(k)$  = Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement, and clear the flag by writing 1.

Time measurement ( $\Delta T$ ) between unit position events is 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.

If the QEP capture timer overflows between unit position events, then the timer sets the QEP capture overflow flag (QEPSTS[COEF]) in the status register and the QCPRDLAT register is set to 0xFFFF. If direction change occurs between the unit position events, then the error flag is set in the status register (QEPSTS[CDEF]) and the QCPRDLAT register is set to 0xFFFF.

The Capture Timer (QCTMR) and Capture Period register (QCPRD) can be configured to latch on the 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 15-19](#) shows the capture unit operation along with the position counter.

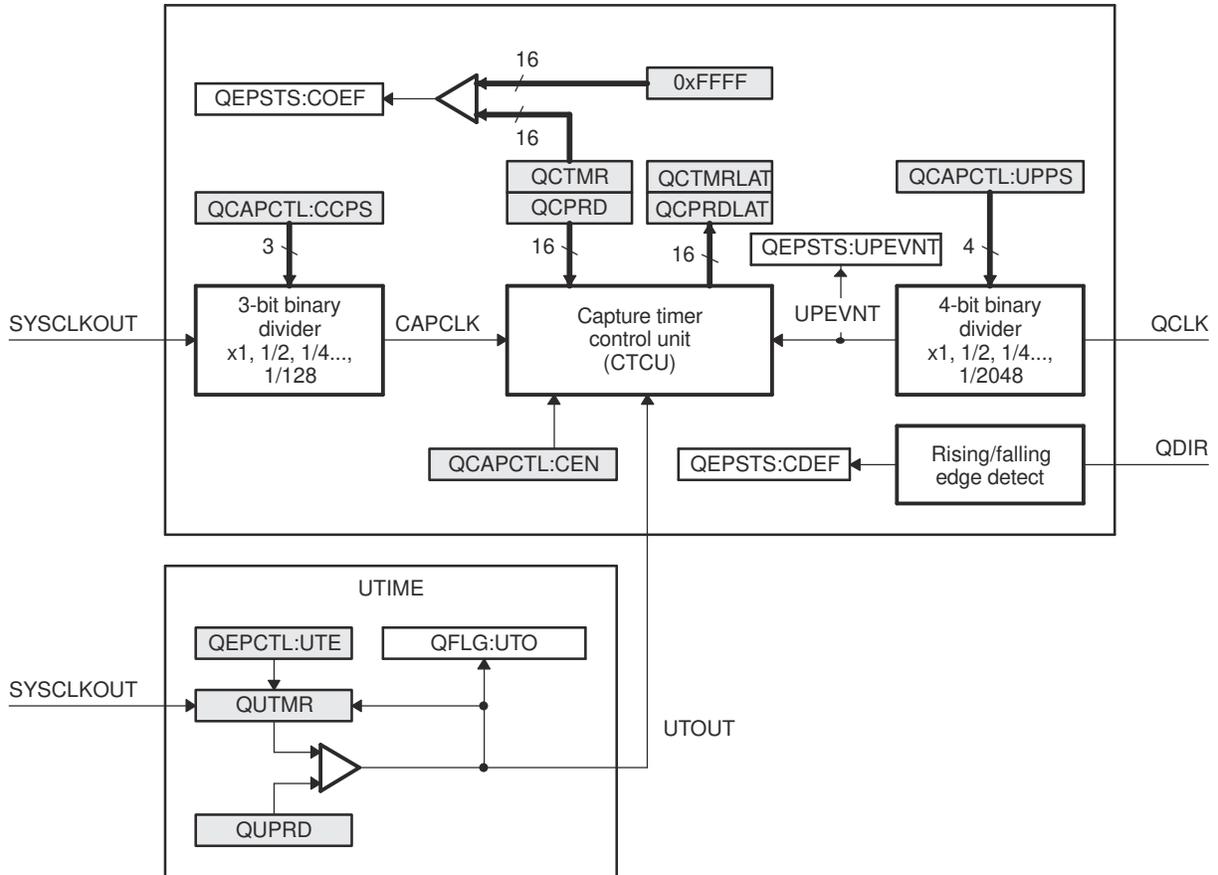
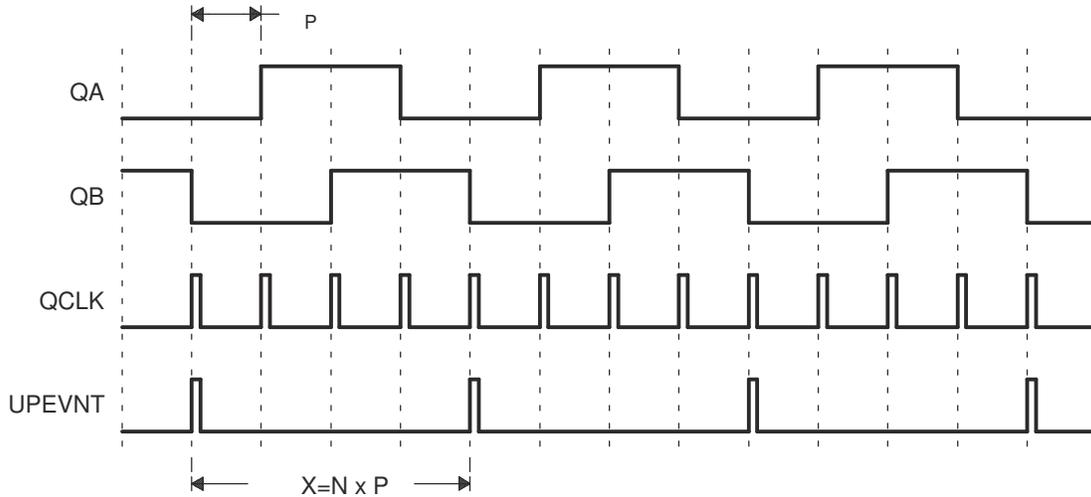


Figure 15-17. eQEP Edge Capture Unit

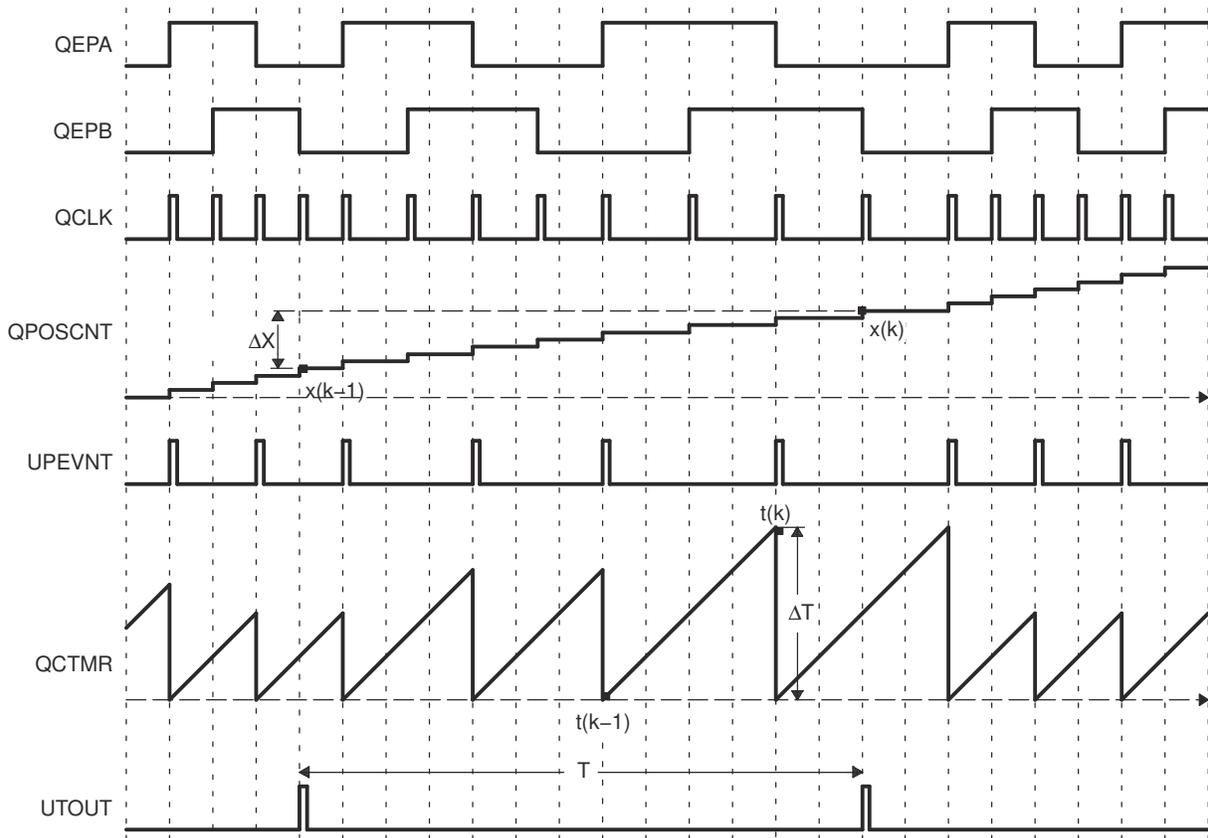
**CAUTION**

The QCAPCTL[UPPS] prescaler cannot be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so can result in undefined behavior. The QCAPCTL[CCPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.



N = Number of quadrature periods selected using QCAPCTL[UPPS] bits

**Figure 15-18. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)**



**Figure 15-19. eQEP Edge Capture Unit - Timing Details**

Velocity calculation equation:

$$v(k) = \frac{x(k) - x(k - 1)}{T} = \frac{\Delta X}{T} \quad (14)$$

where:

- $v(k)$  = Velocity at time instant  $k$
- $x(k)$  = Position at time instant  $k$
- $x(k-1)$  = Position at time instant  $k-1$
- $T$  = Fixed unit time or inverse of velocity calculation rate
- $\Delta X$  = Incremental position movement in unit time
- $X$  = Fixed unit position
- $\Delta T$  = Incremental time elapsed for unit position movement
- $t(k)$  = Time instant "k"
- $t(k-1)$  = Time instant "k-1"

Unit time ( $T$ ) and unit period ( $X$ ) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QOSLAT and QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
$T$	Unit Period Register (QUPRD)
$\Delta X$	Incremental Position = QOSLAT( $k$ ) - QOSLAT( $k-1$ )
$X$	Fixed-unit position defined by sensor resolution and QCAPCTL[UPPS] bits
$\Delta T$	Capture Period Latch (QCPRDLAT)

### 15.7 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer (Figure 15-20) 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 times out and the watchdog interrupt flag is set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

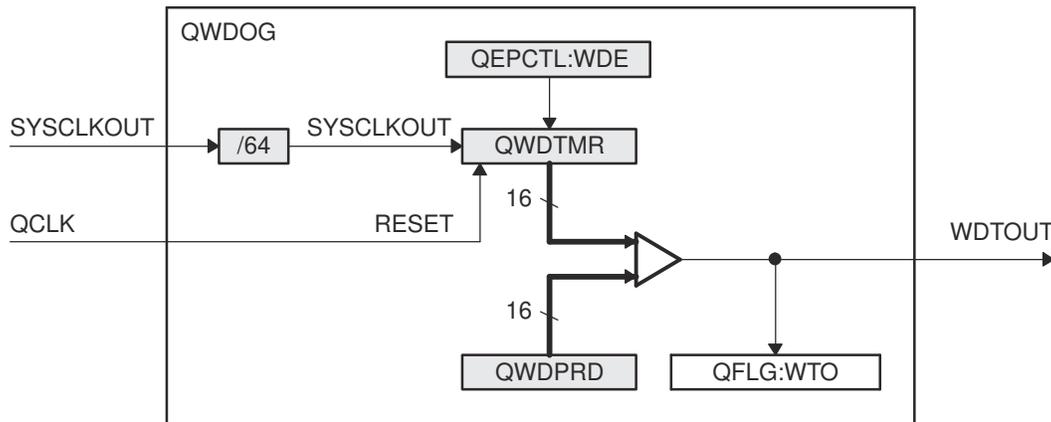


Figure 15-20. eQEP Watchdog Timer

### 15.8 eQEP Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations, see Figure 15-21. Whenever the unit timer (QUTMR) matches the unit period register (QUPRD), the eQEP peripheral resets the unit timer (QUTMR) and also generates the unit time out interrupt flag (QFLG[UTO]). The unit timer gets reset whenever timer value equals to configured period value.

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in Section 15.6.

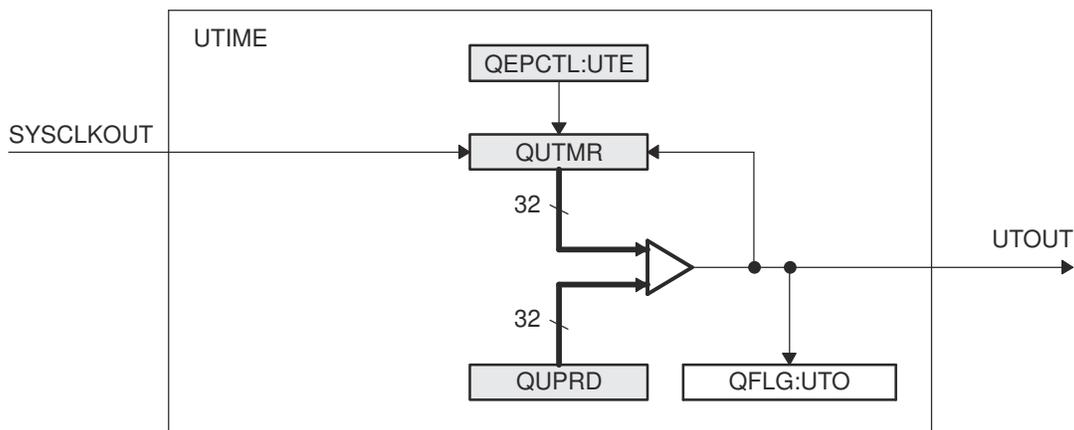


Figure 15-21. eQEP Unit Timer Base

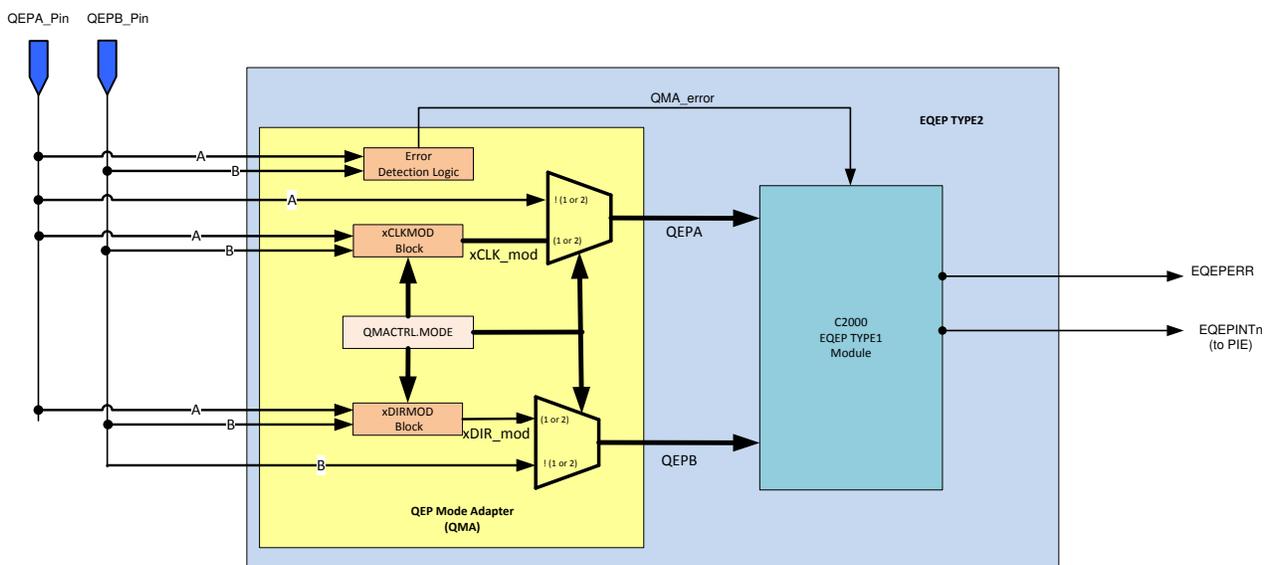
## 15.9 QMA Module

The QEP Mode Adapter (QMA) is designed to extend the C2000™ eQEP module capabilities to support the additional modes described. [Figure 15-22](#) depicts how the QMA module is integrated into the eQEP module.

At reset, by default QMA logic is bypassed and the EQEPA and EQEPB inputs from the pins go directly into the eQEP module. When QMA module is enabled by configuring the QMACTRL[MODE] register, the EQEPA and EQEPB input are processed by this module and modified version of EQEPA and EQEPB signals are sent to the eQEP module. The QMA module requires the eQEP module to be configured in the Direction-Count mode and generates a clock signal on EQEPA input and direction signal on EQEPB input as needed for the proper operation of the intended mode.

- The xCLKMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the clock signal on the EQEPA input to the eQEP module.
- The xDIRMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the direction signal on the EQEPB input to the eQEP module.

The QMA module has error detection logic to detect illegal transitions on EQEPA and EQEPB input signals. The QMA module's error and interrupt are integrated inside the eQEP module as described in [Section 15.10](#). In addition, the QMACTRL register configuration can be locked using the QMALOCK register. Refer to the register description for more details.



**Figure 15-22. QMA Module Block Diagram**

### 15.9.1 Modes of Operation

The QMA module can be operated in the following modes by configuring the QMACTRL register:

- QMA Mode-1 (QMACTRL[MODE]=1)
- QMA Mode-2 (QMACTRL[MODE]=2)

#### 15.9.1.1 QMA Mode-1 (QMACTRL[MODE]=1)

This mode is used when the default state of EQEPA and EQEPB inputs is high. In this mode, outputs of QMA correspond to the following as shown in Figure 15-23:

- EQEPA Output of QMA is the AND of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is the direction signal generated by QMA based on EQEPA and EQEPB inputs

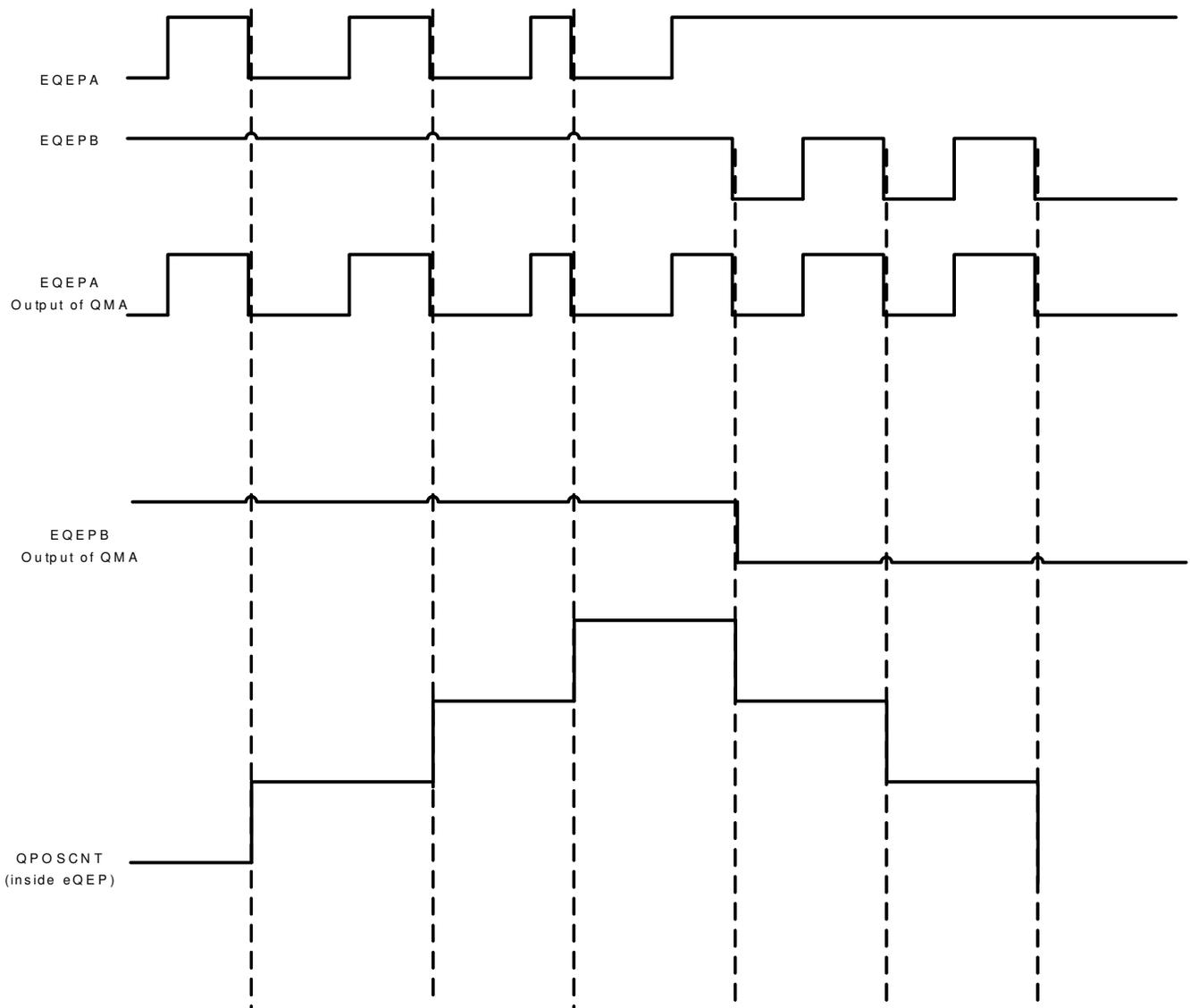


Figure 15-23. QMA Mode-1

### 15.9.1.2 QMA Mode-2 (QMACTRL[MODE]=2)

This mode is used when the default state of EQEPA and EQEPB inputs is low. In this mode, outputs of QMA correspond to the following as shown in Figure 15-24:

- EQEPA Output of QMA is the OR of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is the direction signal generated by QMA based on EQEPA and EQEPB inputs

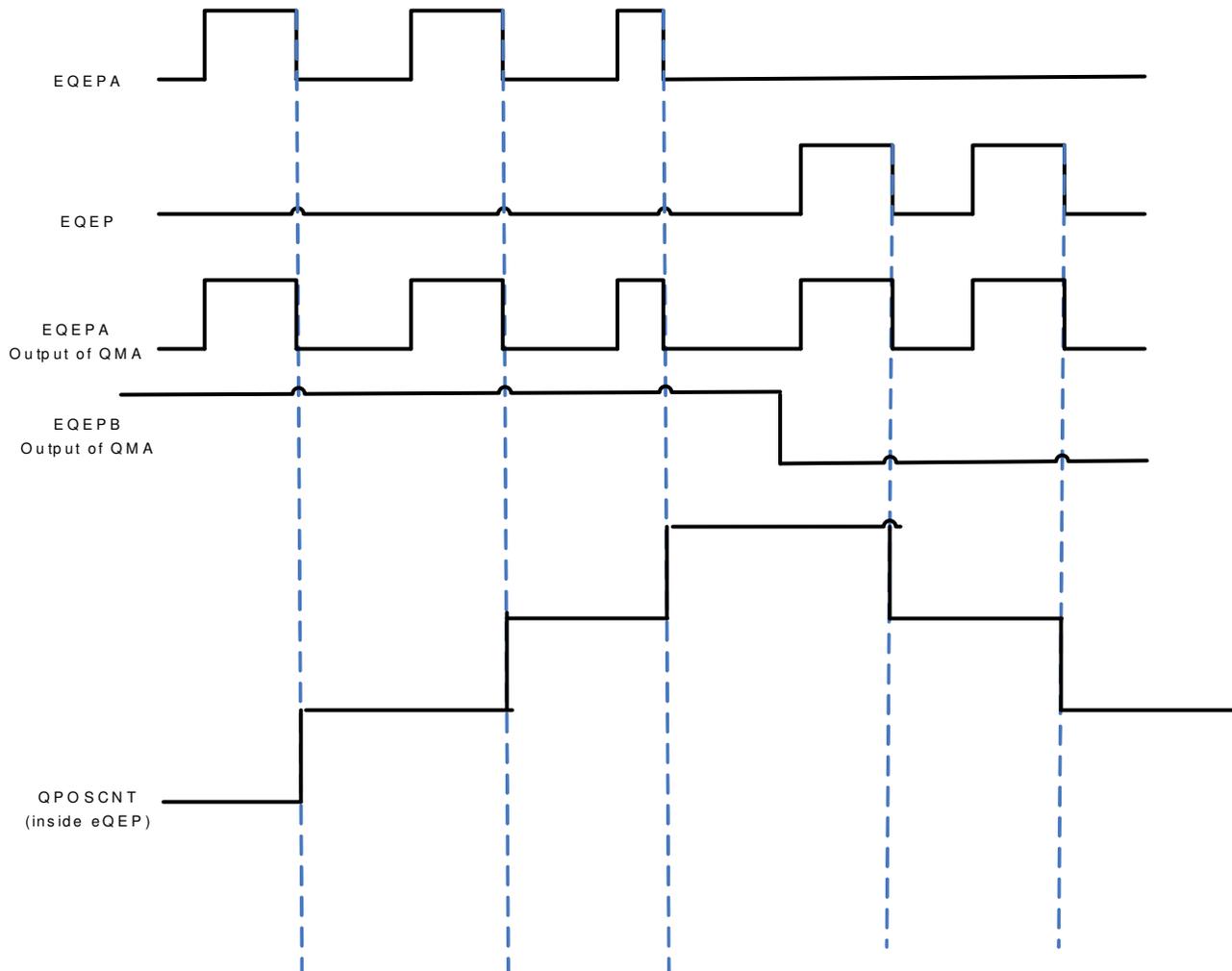


Figure 15-24. QMA Mode-2

### 15.9.2 Interrupt and Error Generation

The error detection logic detects illegal transitions on EQEPA and EQEPB signals and generates an error signal. This error signal can be used to generate eQEP interrupt and error output. Refer to Section 15.10 for details.

### 15.10 eQEP Interrupt Structure

Figure 15-25 shows how the interrupt mechanism works in the eQEP module.

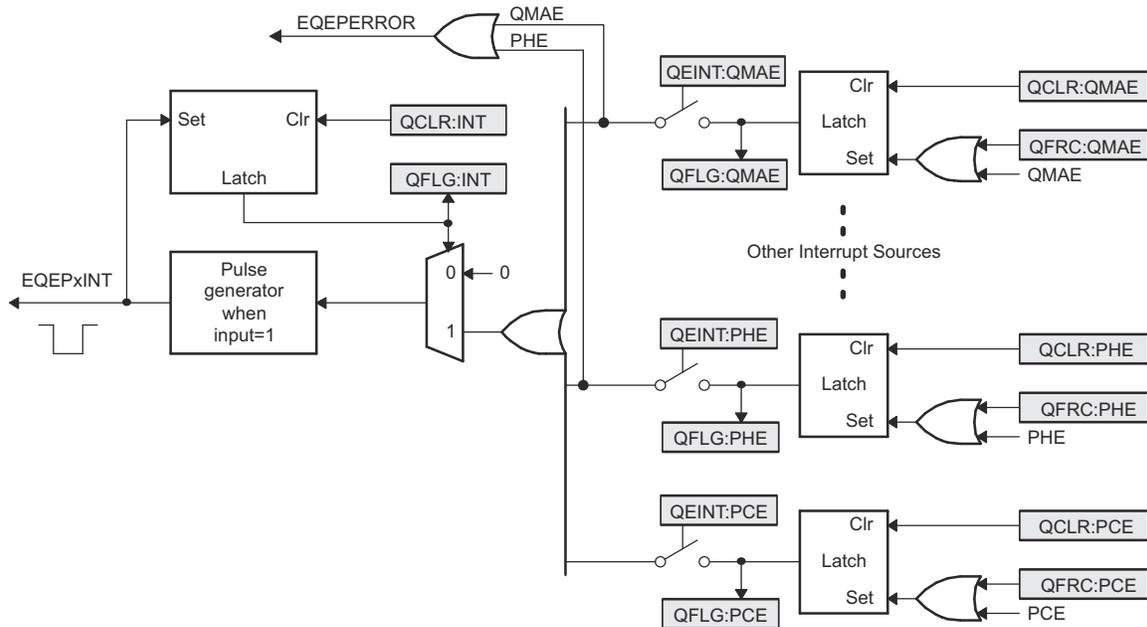


Figure 15-25. eQEP Interrupt Generation

Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT).

An interrupt pulse is generated to PIE when:

1. Interrupt is enabled for eQEP event inside QEINT register
2. Interrupt flag for eQEP event inside QFLG register is set, and
3. Global interrupt status flag bit QFLG[INT] had been cleared for previously generated interrupt event. The interrupt service routine needs to clear the global interrupt flag bit and the serviced event, by way of the interrupt clear register (QCLR), before any other interrupt pulses are generated. If either flags inside the QFLG register are not cleared, further interrupt events do not generate an interrupt to PIE. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

## 15.11 Software

### 15.11.1 EQEP Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location: C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/eqep

Cloud access to these examples is available at the following link: [dev.ti.com C2000Ware Examples](https://dev.ti.com/C2000Ware/Examples).

#### 15.11.1.1 Frequency Measurement Using eQEP

FILE: eqep\_ex1\_freq\_cal.c

This example will calculate the frequency of an input signal using the eQEP module. ePWM1A is configured to generate this input signal with a frequency of 5 kHz. It will interrupt once every period and call the frequency calculation function. This example uses the IQMath library to simplify high-precision calculations.

In addition to the main example file, the following files must be included in this project:

- *eqep\_ex1\_calculation.c* - contains frequency calculation function
- *eqep\_ex1\_calculation.h* - includes initialization values for frequency structure

The configuration for this example is as follows

- Maximum frequency is configured to 10KHz (baseFreq)
- Minimum frequency is assumed at 50Hz for capture pre-scalar selection

*SPEED\_FR*: High Frequency Measurement is obtained by counting the external input pulses for 10ms (unit timer set to 100Hz).  $\text{SPEED\_FR} = \frac{\text{Count} \Delta}{10\text{ms}}$

*SPEED\_PR*: Low Frequency Measurement is obtained by measuring time period of input edges. Time measurement is averaged over 64 edges for better results and the capture unit performs the time measurement using pre-scaled SYSCLK.

Note that the pre-scalar for capture unit clock is selected such that the capture timer does not overflow at the required minimum frequency. This example runs indefinitely until the user stops it.

For more information about the frequency calculation see the comments at the beginning of eqep\_ex1\_calculation.c and the XLS file provided with the project, eqep\_ex1\_calculation.xls.

- For External connections, Control Card settings are used by default. To use launchpad pins for eQEP1A select them in SysConfig.

*ExternalConnections* for Control Card

- Connect GPIO6/eQEP1A to GPIO0/ePWM1A

#### 15.11.1.2 Position and Speed Measurement Using eQEP

FILE: eqep\_ex2\_pos\_speed.c

This example provides position and speed measurement using the capture unit and speed measurement using unit time out of the eQEP module. ePWM1 and a GPIO are configured to generate simulated eQEP signals. The ePWM module will interrupt once every period and call the position/speed calculation function. This example uses the IQMath library to simplify high-precision calculations.

In addition to the main example file, the following files must be included in this project:

- *eqep\_ex2\_calculation.c* - contains position/speed calculation function
- *eqep\_ex2\_calculation.h* - includes initialization values for position/speed structure

The configuration for this example is as follows

- Maximum speed is configured to 6000rpm (baseRPM)
- Minimum speed is assumed at 10rpm for capture pre-scalar selection
- Pole pair is configured to 2 (polePairs)
- Encoder resolution is configured to 4000 counts/revolution (mechScaler)

- Which means:  $4000 / 4 = 1000$  line/revolution quadrature encoder (simulated by ePWM1)
- ePWM1 (simulating QEP encoder signals) is configured for a 5kHz frequency or 300 rpm ( $= 4 * 5000 \text{ cnts/sec} * 60 \text{ sec/min} / 4000 \text{ cnts/rev}$ )

**SPEEDRPM\_FR:** High Speed Measurement is obtained by counting the QEP input pulses for 10ms (unit timer set to 100Hz).

$$\text{SPEEDRPM\_FR} = (\text{Position Delta} / 10\text{ms}) * 60 \text{ rpm}$$

**SPEEDRPM\_PR:** Low Speed Measurement is obtained by measuring time period of QEP edges. Time measurement is averaged over 64 edges for better results and the capture unit performs the time measurement using pre-scaled SYSCLK.

Note that the pre-scaler for capture unit clock is selected such that the capture timer does not overflow at the required minimum frequency. This example runs indefinitely until the user stops it.

For more information about the position/speed calculation see the comments at the beginning of `eqep_ex2_calculation.c` and the XLS file provided with the project, `eqep_ex2_calculation.xls`.

#### *External Connections*

- Connect GPIO6/eQEP1A to GPIO0/ePWM1A (simulates eQEP Phase A signal)
- Connect GPIO7/eQEP1B to GPIO1/ePWM1B (simulates eQEP Phase B signal)
- Connect GPIO9/eQEP1I to GPIO4 (simulates eQEP Index Signal)

#### *Watch Variables*

- `posSpeed.speedRPMFR` - Speed meas. in rpm using QEP position counter
- `posSpeed.speedRPMPR` - Speed meas. in rpm using capture unit
- `posSpeed.thetaMech` - Motor mechanical angle (Q15)
- `posSpeed.thetaElec` - Motor electrical angle (Q15)

### **15.11.1.3 ePWM frequency Measurement Using eQEP via xbar connection**

FILE: `eqep_ex3_epwm_xbar.c`

This example will calculate the frequency of an PWM signal using the eQEP module. ePWM1A is configured to generate this input signal with a frequency of 5 kHz. This ePWM signal is connected to input of eQEP using Input CrossBar and EPWM XBAR. ePWM module will interrupt once every period and call the frequency calculation function. This example uses the IQMath library to simplify high-precision calculations.

In addition to the main example file, the following files must be included in this project:

- `eqep_ex1_calculation.c` - contains frequency calculation function
- `eqep_ex1_calculation.h` - includes initialization values for frequency structure

The configuration for this example is as follows

- Maximum frequency is configured to 10KHz (baseFreq)
- Minimum frequency is assumed at 50Hz for capture pre-scalar selection
- GPIO0 is connected to output of INPUT\_XBAR1
- INPUT\_XBAR1 is connected to output of PWMXBAR at TRIP4
- eQEPA source is configured as PWMXBAR.1 output (TRIP4)

**SPEED\_FR:** High Frequency Measurement is obtained by counting the external input pulses for 10ms (unit timer set to 100Hz).  $\lfloor \text{SPEED\_FR} = \frac{\text{Count} \Delta}{10\text{ms}} \rfloor$

**SPEED\_PR:** Low Frequency Measurement is obtained by measuring time period of input edges. Time measurement is averaged over 64 edges for better results and the capture unit performs the time measurement using pre-scaled SYSCLK.

Note that the pre-scaler for capture unit clock is selected such that the capture timer does not overflow at the required minimum frequency. This example runs indefinitely until the user stops it.

For more information about the frequency calculation see the comments at the beginning of `eqep_ex1_calculation.c` and the XLS file provided with the project, `eqep_ex1_calculation.xls`.

#### *Watch Variables*

- `freq.freqHzFR` - Frequency measurement using position counter/unit time out
- `freq.freqHzPR` - Frequency measurement using capture unit

#### **15.11.1.4 Frequency Measurement Using eQEP via unit timeout interrupt**

FILE: `eqep_ex4_freq_cal_interrupt.c`

This example will calculate the frequency of an input signal using the eQEP module. ePWM1A is configured to generate this input signal with a frequency of 5 kHz. EQEP unit timeout is set which will generate an interrupt every `UNIT_PERIOD` microseconds and frequency calculation occurs continuously

The configuration for this example is as follows

- PWM frequency is specified as 5000Hz
- `UNIT_PERIOD` is specified as 10000 us
- Min frequency is  $(1/(2*10\text{ms}))$  i.e 50Hz
- Highest frequency can be  $(2^{32}/((2*10\text{ms}))$
- Resolution of frequency measurement is 50hz

`freq` : Frequency Measurement is obtained by counting the external input pulses for `UNIT_PERIOD` (unit timer set to 10 ms).

- *ExternalConnections* for Control Card
- Connect GPIO6/eQEP1A to GPIO0/ePWM1A

#### *Watch Variables*

- `freq` - Frequency measurement using position counter/unit time out
- `pass` - If measured frequency matches with PWM frequency then `pass` = 1 else 0

#### **15.11.1.5 Motor speed and direction measurement using eQEP via unit timeout interrupt**

FILE: `eqep_ex5_speed_dir_motor.c`

This example can be used to sense the speed and direction of motor using eQEP in quadrature encoder mode. ePWM1A is configured to simulate motor encoder signals with frequency of 5 kHz on both A and B pins with 90 degree phase shift (so as to run this example without motor). EQEP unit timeout is set which will generate an interrupt every `UNIT_PERIOD` microseconds and speed calculation occurs continuously based on the direction of motor

The configuration for this example is as follows

- PWM frequency is specified as 5000Hz
- `UNIT_PERIOD` is specified as 10000 us
- Simulated quadrature signal frequency is 20000Hz ( $4 * 5000$ )
- Encoder holes assumed as 1000
- Thus Simulated motor speed is 300rpm ( $5000 * (60 / 1000)$ )

`freq` : Simulated quadrature signal frequency measured by counting the external input pulses for `UNIT_PERIOD` (unit timer set to 10 ms). `speed` : Measure motor speed in rpm `dir` : Indicates clockwise (1) or anticlockwise (-1)

*External Connections* (if motor encoder signals are simulated by ePWM)

- Connect GPIO6/eQEP1A to GPIO0/ePWM1A (simulates eQEP Phase A signal)
- Connect GPIO7/eQEP1B to GPIO1/ePWM1B (simulates eQEP Phase B signal)

#### *Watch Variables*

- *freq* : Simulated motor frequency measurement is obtained by counting the external input pulses for UNIT\_PERIOD (unit timer set to 10 ms).
- *speed* : Measure motor speed in rpm
- *dir* : Indicates clockwise (1) or anticlockwise (-1)
- *pass* - If measured quadrature frequency matches with i.e. input quadrature frequency (4 \* PWM frequency) then pass = 1 else fail = 1 (\*\* only when "MOTOR" is commented out)

## 15.12 eQEP Registers

This section describes the Enhanced Quadrature Encoder Pulse Registers.

### 15.12.1 EQEP Base Address Table

**Table 15-4. EQEP Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
EQep1Regs	EQEP_REGS	EQEP1_BASE	0x0000_5100	YES

## 15.12.2 EQEP\_REGS Registers

Table 15-5 lists the memory-mapped registers for the EQEP\_REGS registers. All register offset addresses not listed in Table 15-5 should be considered as reserved locations and the register contents should not be modified.

**Table 15-5. EQEP\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	QPOSCNT	Position Counter		<a href="#">Go</a>
2h	QPOSINIT	Position Counter Init		<a href="#">Go</a>
4h	QPOSMAX	Maximum Position Count		<a href="#">Go</a>
6h	QPOSCMP	Position Compare		<a href="#">Go</a>
8h	QPOSILAT	Index Position Latch		<a href="#">Go</a>
Ah	QPOSSLAT	Strobe Position Latch		<a href="#">Go</a>
Ch	QPOSLAT	Position Latch		<a href="#">Go</a>
Eh	QUTMR	QEP Unit Timer		<a href="#">Go</a>
10h	QUPRD	QEP Unit Period		<a href="#">Go</a>
12h	QWDTMR	QEP Watchdog Timer		<a href="#">Go</a>
13h	QWDPRD	QEP Watchdog Period		<a href="#">Go</a>
14h	QDECCTL	Quadrature Decoder Control		<a href="#">Go</a>
15h	QEPCTL	QEP Control		<a href="#">Go</a>
16h	QCAPCTL	Quadrature Capture Control		<a href="#">Go</a>
17h	QPOSCTL	Position Compare Control		<a href="#">Go</a>
18h	QEINT	QEP Interrupt Control		<a href="#">Go</a>
19h	QFLG	QEP Interrupt Flag		<a href="#">Go</a>
1Ah	QCLR	QEP Interrupt Clear		<a href="#">Go</a>
1Bh	QFRC	QEP Interrupt Force		<a href="#">Go</a>
1Ch	QEPSTS	QEP Status		<a href="#">Go</a>
1Dh	QCTMR	QEP Capture Timer		<a href="#">Go</a>
1Eh	QCPRD	QEP Capture Period		<a href="#">Go</a>
1Fh	QCTMRLAT	QEP Capture Latch		<a href="#">Go</a>
20h	QCPRLAT	QEP Capture Period Latch		<a href="#">Go</a>
30h	REV	QEP Revision Number		<a href="#">Go</a>
32h	QEPSTROBESEL	QEP Strobe select register		<a href="#">Go</a>
34h	QMACTRL	QMA Control register		<a href="#">Go</a>
36h	QEPRCSEL	QEP Source Select Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 15-6 shows the codes that are used for access types in this section.

**Table 15-6. EQEP\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear

**Table 15-6. EQEP\_REGS Access Type Codes (continued)**

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

### 15.12.2.1 QPOSCNT Register (Offset = 0h) [Reset = 0000000h]

QPOSCNT is shown in [Figure 15-26](#) and described in [Table 15-7](#).

Return to the [Summary Table](#).

Position Counter

**Figure 15-26. QPOSCNT Register**

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

**Table 15-7. QPOSCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSCNT	R/W	0h	Position Counter This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point. This Register acts as a Read ONLY register while counter is counting up/down. Note: It is recommended to only write to the position counter register (QPOSCNT) during initialization, i.e. when the eQEP position counter is disabled (QPEN bit of QEPCTL is zero). Once the position counter is enabled (QPEN bit is one), writing to the eQEP position counter register (QPOSCNT) may cause unexpected results. Reset type: SYSRSn

### 15.12.2.2 QPOSINIT Register (Offset = 2h) [Reset = 0000000h]

QPOSINIT is shown in [Figure 15-27](#) and described in [Table 15-8](#).

Return to the [Summary Table](#).

Position Counter Init

**Figure 15-27. QPOSINIT Register**

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

**Table 15-8. QPOSINIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSINIT	R/W	0h	Position Counter Init This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter can be initialized through software. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

### 15.12.2.3 QPOSMAX Register (Offset = 4h) [Reset = 00000000h]

QPOSMAX is shown in [Figure 15-28](#) and described in [Table 15-9](#).

Return to the [Summary Table](#).

Maximum Position Count

**Figure 15-28. QPOSMAX Register**

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

**Table 15-9. QPOSMAX Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSMAX	R/W	0h	Maximum Position Count This register contains the maximum position counter value. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

### 15.12.2.4 QPOSCMP Register (Offset = 6h) [Reset = 00000000h]

QPOSCMP is shown in [Figure 15-29](#) and described in [Table 15-10](#).

Return to the [Summary Table](#).

Position Compare

**Figure 15-29. QPOSCMP Register**

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

**Table 15-10. 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. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

### 15.12.2.5 QPOSILAT Register (Offset = 8h) [Reset = 00000000h]

QPOSILAT is shown in [Figure 15-30](#) and described in [Table 15-11](#).

Return to the [Summary Table](#).

Index Position Latch

**Figure 15-30. QPOSILAT Register**

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

**Table 15-11. QPOSILAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSILAT	R	0h	Index Position Latch The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits. Reset type: SYSRSn

### 15.12.2.6 QPOSSLAT Register (Offset = Ah) [Reset = 00000000h]

QPOSSLAT is shown in [Figure 15-31](#) and described in [Table 15-12](#).

Return to the [Summary Table](#).

Strobe Position Latch

**Figure 15-31. 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																															

**Table 15-12. QPOSSLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSSLAT	R	0h	Strobe Position Latch The position-counter value is latched into this register on a strobe event as defined by the QEPCTL[SEL] bits. Reset type: SYSRSn

### 15.12.2.7 QPOSLAT Register (Offset = Ch) [Reset = 0000000h]

QPOSLAT is shown in [Figure 15-32](#) and described in [Table 15-13](#).

Return to the [Summary Table](#).

Position Latch

**Figure 15-32. QPOSLAT Register**

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

**Table 15-13. QPOSLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QPOSLAT	R	0h	Position Latch The position-counter value is latched into this register on a unit time out event. Reset type: SYSRSn

### 15.12.2.8 QUTMR Register (Offset = Eh) [Reset = 0000000h]

QUTMR is shown in [Figure 15-33](#) and described in [Table 15-14](#).

Return to the [Summary Table](#).

QEP Unit Timer

**Figure 15-33. QUTMR Register**

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

**Table 15-14. 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 the unit time period value a unit time event is generated. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

### 15.12.2.9 QUPRD Register (Offset = 10h) [Reset = 0000000h]

QUPRD is shown in [Figure 15-34](#) and described in [Table 15-15](#).

Return to the [Summary Table](#).

QEP Unit Period

**Figure 15-34. QUPRD Register**

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

**Table 15-15. QUPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	QUPRD	R/W	0h	QEP Unit Period This register contains the period count for the unit timer to generate periodic unit time events. These events latch the eQEP position information at periodic intervals and optionally generate an interrupt. Writes to this register should always be full 32-bit writes. Reset type: SYSRSn

**15.12.2.10 QWDTMR Register (Offset = 12h) [Reset = 0000h]**

QWDTMR is shown in [Figure 15-35](#) and described in [Table 15-16](#).

Return to the [Summary Table](#).

QEP Watchdog Timer

**Figure 15-35. QWDTMR Register**

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

**Table 15-16. QWDTMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QWDTMR	R/W	0h	QEP Watchdog Timer This register acts as time base for the watchdog to detect motor stalls. When this timer value matches with the watchdog's period value a watchdog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion. Reset type: SYSRSn

### 15.12.2.11 QWDPRD Register (Offset = 13h) [Reset = 0000h]

QWDPRD is shown in [Figure 15-36](#) and described in [Table 15-17](#).

Return to the [Summary Table](#).

QEP Watchdog Period

**Figure 15-36. QWDPRD Register**

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

**Table 15-17. QWDPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QWDPRD	R/W	0h	QEP Watchdog Period This register contains the time-out count for the eQEP peripheral watch dog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated. Reset type: SYSRSn

### 15.12.2.12 QDECCTL Register (Offset = 14h) [Reset = 0000h]

QDECCTL is shown in [Figure 15-37](#) and described in [Table 15-18](#).

Return to the [Summary Table](#).

Quadrature Decoder Control

**Figure 15-37. 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				QIDIRE
R/W-0h	R/W-0h	R/W-0h	R-0h				R/W-0h

**Table 15-18. QDECCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	QSRC	R/W	0h	Position-counter source selection Reset type: SYSRSn 0h (R/W) = Quadrature count mode (QCLK = iCLK, QDIR = iDIR) 1h (R/W) = Direction-count mode (QCLK = xCLK, QDIR = xDIR) 2h (R/W) = UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1) 3h (R/W) = DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)
13	SOEN	R/W	0h	Sync output-enable Reset type: SYSRSn 0h (R/W) = Disable position-compare sync output 1h (R/W) = Enable position-compare sync output
12	SPSEL	R/W	0h	Sync output pin selection Reset type: SYSRSn 0h (R/W) = Index pin is used for sync output 1h (R/W) = Strobe pin is used for sync output
11	XCR	R/W	0h	External Clock Rate Reset type: SYSRSn 0h (R/W) = 2x resolution: Count the rising/falling edge 1h (R/W) = 1x resolution: Count the rising edge only
10	SWAP	R/W	0h	CLK/DIR Signal Source for Position Counter Reset type: SYSRSn 0h (R/W) = Quadrature-clock inputs are not swapped 1h (R/W) = Quadrature-clock inputs are swapped
9	IGATE	R/W	0h	Index pulse gating option Reset type: SYSRSn 0h (R/W) = Disable gating of Index pulse 1h (R/W) = Gate the index pin with strobe
8	QAP	R/W	0h	QEPA input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPA input
7	QBP	R/W	0h	QEPB input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPB input
6	QIP	R/W	0h	QEPI input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPI input

**Table 15-18. QDECCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	QSP	R/W	0h	QEPS input polarity Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Negates QEPS input
4-1	RESERVED	R	0h	Reserved
0	QIDIRE	R/W	0h	0 - Compatible mode, Behavior same as existing devices 1 - Enhancement for Direction change during Index will be enabled: On QEPI direction change, the incoming posedge of QA can erroneously update/reset the position counter of the eQEP. This bit only needs to be enabled if the application requires a direction change occurring at the same time as an incoming QEPI signal, or when erroneous PC resets are observed. Reset type: SYSRSn

### 15.12.2.13 QEPCNTL Register (Offset = 15h) [Reset = 0000h]

QEPCNTL is shown in [Figure 15-38](#) and described in [Table 15-19](#).

Return to the [Summary Table](#).

QEP Control

**Figure 15-38. QEPCNTL Register**

15	14	13	12	11	10	9	8
FREE_SOFT		PCRM		SEI		IEI	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
SWI	SEL	IEL		QPEN	QCLM	UTE	WDE
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 15-19. QEPCNTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation mode Reset type: SYSRSn 0h (R/W) = QPOSCNT behavior Position counter stops immediately on emulation suspend 0h (R/W) = QWDTMR behavior Watchdog counter stops immediately 0h (R/W) = QUTMR behavior Unit timer stops immediately 0h (R/W) = QCTMR behavior Capture Timer stops immediately 1h (R/W) = QPOSCNT behavior Position counter continues to count until the rollover 1h (R/W) = QWDTMR behavior Watchdog counter counts until WD period match roll over 1h (R/W) = QUTMR behavior Unit timer counts until period rollover 1h (R/W) = QCTMR behavior Capture Timer counts until next unit period event 2h (R/W) = QPOSCNT behavior Position counter is unaffected by emulation suspend 2h (R/W) = QWDTMR behavior Watchdog counter is unaffected by emulation suspend 2h (R/W) = QUTMR behavior Unit timer is unaffected by emulation suspend 2h (R/W) = QCTMR behavior Capture Timer is unaffected by emulation suspend 3h (R/W) = Same as FREE_SOFT_2
13-12	PCRM	R/W	0h	Position counter reset Reset type: SYSRSn 0h (R/W) = Position counter reset on an index event 1h (R/W) = Position counter reset on the maximum position 2h (R/W) = Position counter reset on the first index event 3h (R/W) = Position counter reset on a unit time event
11-10	SEI	R/W	0h	Strobe event initialization of position counter Reset type: SYSRSn 0h (R/W) = Does nothing (action disabled) 1h (R/W) = Does nothing (action disabled) 2h (R/W) = Initializes the position counter on rising edge of the QEPS signal 3h (R/W) = Clockwise Direction: Initializes the position counter on the rising edge of QEPS strobe Counter Clockwise Direction: Initializes the position counter on the falling edge of QEPS strobe

**Table 15-19. QEPCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9-8	IEI	R/W	0h	Index event init of position count Reset type: SYSRSn 0h (R/W) = Do nothing (action disabled) 1h (R/W) = Do nothing (action disabled) 2h (R/W) = Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT) 3h (R/W) = Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT)
7	SWI	R/W	0h	Software init position counter Reset type: SYSRSn 0h (R/W) = Do nothing (action disabled) 1h (R/W) = Initialize position counter (QPOSCNT=QPOSINIT). This bit is not cleared automatically
6	SEL	R/W	0h	Strobe event latch of position counter Reset type: SYSRSn 0h (R/W) = The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCCNT). Latching on the falling edge can be done by inverting the strobe input using the QSP bit in the QDECCTL register 1h (R/W) = Clockwise Direction: Position counter is latched on rising edge of QEPS strobe Counter Clockwise Direction: Position counter is latched on falling edge of QEPS strobe
5-4	IEL	R/W	0h	Index event latch of position counter (software index marker) Reset type: SYSRSn 0h (R/W) = Reserved 1h (R/W) = Latches position counter on rising edge of the index signal 2h (R/W) = Latches position counter on falling edge of the index signal 3h (R/W) = Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking.
3	QPEN	R/W	0h	Quadrature position counter enable/software reset Reset type: SYSRSn 0h (R/W) = Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset. When QPEN is disabled, some flags in the QFLG register do not get reset or cleared and show the actual state of that flag. 1h (R/W) = eQEP position counter is enabled
2	QCLM	R/W	0h	QEP capture latch mode Reset type: SYSRSn 0h (R/W) = Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register. 1h (R/W) = Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSILAT, QCTMRLAT and QCPRDLAT registers on unit time out.
1	UTE	R/W	0h	QEP unit timer enable Reset type: SYSRSn 0h (R/W) = Disable eQEP unit timer 1h (R/W) = Enable unit timer
0	WDE	R/W	0h	QEP watchdog enable Reset type: SYSRSn 0h (R/W) = Disable the eQEP watchdog timer 1h (R/W) = Enable the eQEP watchdog timer

### 15.12.2.14 QCAPCTL Register (Offset = 16h) [Reset = 0000h]

QCAPCTL is shown in [Figure 15-39](#) and described in [Table 15-20](#).

Return to the [Summary Table](#).

Quadrature Capture Control

**Figure 15-39. QCAPCTL Register**

15	14	13	12	11	10	9	8
CEN		RESERVED					
R/W-0h				R-0h			
7	6	5	4	3	2	1	0
RESERVED		CCPS			UPPS		
R-0h		R/W-0h			R/W-0h		

**Table 15-20. QCAPCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	CEN	R/W	0h	Enable eQEP capture Reset type: SYSRSn 0h (R/W) = eQEP capture unit is disabled 1h (R/W) = eQEP capture unit is enabled
14-7	RESERVED	R	0h	Reserved
6-4	CCPS	R/W	0h	eQEP capture timer clock prescaler Reset type: SYSRSn 0h (R/W) = CAPCLK = SYSCLKOUT/1 1h (R/W) = CAPCLK = SYSCLKOUT/2 2h (R/W) = CAPCLK = SYSCLKOUT/4 3h (R/W) = CAPCLK = SYSCLKOUT/8 4h (R/W) = CAPCLK = SYSCLKOUT/16 5h (R/W) = CAPCLK = SYSCLKOUT/32 6h (R/W) = CAPCLK = SYSCLKOUT/64 7h (R/W) = CAPCLK = SYSCLKOUT/128
3-0	UPPS	R/W	0h	Unit position event prescaler Reset type: SYSRSn 0h (R/W) = UPEVNT = QCLK/1 1h (R/W) = UPEVNT = QCLK/2 2h (R/W) = UPEVNT = QCLK/4 3h (R/W) = UPEVNT = QCLK/8 4h (R/W) = UPEVNT = QCLK/16 5h (R/W) = UPEVNT = QCLK/32 6h (R/W) = UPEVNT = QCLK/64 7h (R/W) = UPEVNT = QCLK/128 8h (R/W) = UPEVNT = QCLK/256 9h (R/W) = UPEVNT = QCLK/512 Ah (R/W) = UPEVNT = QCLK/1024 Bh (R/W) = UPEVNT = QCLK/2048 Ch (R/W) = Reserved Dh (R/W) = Reserved Eh (R/W) = Reserved Fh (R/W) = Reserved

### 15.12.2.15 QPOSCTL Register (Offset = 17h) [Reset = 0000h]

QPOSCTL is shown in [Figure 15-40](#) and described in [Table 15-21](#).

Return to the [Summary Table](#).

Position Compare Control

**Figure 15-40. QPOSCTL Register**

15	14	13	12	11	10	9	8
PCSHDW	PCLOAD	PCPOL	PCE	PCSPW			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
PCSPW							
R/W-0h							

**Table 15-21. QPOSCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	PCSHDW	R/W	0h	Position compare of shadow enable Reset type: SYSRSn 0h (R/W) = Shadow disabled, load Immediate 1h (R/W) = Shadow enabled
14	PCLOAD	R/W	0h	Position compare of shadow load Reset type: SYSRSn 0h (R/W) = Load on QPOSCNT = 0 1h (R/W) = Load when QPOSCNT = QPOSCMP
13	PCPOL	R/W	0h	Polarity of sync output Reset type: SYSRSn 0h (R/W) = Active HIGH pulse output 1h (R/W) = Active LOW pulse output
12	PCE	R/W	0h	Position compare enable/disable Reset type: SYSRSn 0h (R/W) = Disable position compare unit 1h (R/W) = Enable position compare unit
11-0	PCSPW	R/W	0h	Select-position-compare sync output pulse width Reset type: SYSRSn 0h (R/W) = 1 * 4 * SYSCLKOUT cycles 1h (R/W) = 2 * 4 * SYSCLKOUT cycles FFFh (R/W) = 4096 * 4 * SYSCLKOUT cycles

### 15.12.2.16 QEINT Register (Offset = 18h) [Reset = 0000h]

QEINT is shown in [Figure 15-41](#) and described in [Table 15-22](#).

Return to the [Summary Table](#).

QEP Interrupt Control

**Figure 15-41. QEINT Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	QPE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 15-22. QEINT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R/W	0h	QMA Error Interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
11	UTO	R/W	0h	Unit time out interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
10	IEL	R/W	0h	Index event latch interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
9	SEL	R/W	0h	Strobe event latch interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
8	PCM	R/W	0h	Position-compare match interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
7	PCR	R/W	0h	Position-compare ready interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
6	PCO	R/W	0h	Position counter overflow interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
5	PCU	R/W	0h	Position counter underflow interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
4	WTO	R/W	0h	Watchdog time out interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled

**Table 15-22. QEINT Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	QDC	R/W	0h	Quadrature direction change interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
2	QPE	R/W	0h	Quadrature phase error interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
1	PCE	R/W	0h	Position counter error interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt is disabled 1h (R/W) = Interrupt is enabled
0	RESERVED	R	0h	Reserved

### 15.12.2.17 QFLG Register (Offset = 19h) [Reset = 0000h]

QFLG is shown in [Figure 15-42](#) and described in [Table 15-23](#).

Return to the [Summary Table](#).

QEP Interrupt Flag

**Figure 15-42. QFLG Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

**Table 15-23. QFLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R	0h	QMA Error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
11	UTO	R	0h	Unit time out interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set by eQEP unit timer period match
10	IEL	R	0h	Index event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after latching the QPOSCNT to QPOSILAT
9	SEL	R	0h	Strobe event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after latching the QPOSCNT to QPOSSLAT
8	PCM	R	0h	eQEP compare match event interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position-compare match
7	PCR	R	0h	Position-compare ready interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set after transferring the shadow register value to the active position compare register
6	PCO	R	0h	Position counter overflow interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position counter overflow.
5	PCU	R	0h	Position counter underflow interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = This bit is set on position counter underflow.
4	WTO	R	0h	Watchdog timeout interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set by watchdog timeout

**Table 15-23. QFLG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	QDC	R	0h	Quadrature direction change interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated
2	PHE	R	0h	Quadrature phase error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Set on simultaneous transition of QEPA and QEPB
1	PCE	R	0h	Position counter error interrupt flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Position counter error
0	INT	R	0h	Global interrupt status flag Reset type: SYSRSn 0h (R/W) = No interrupt generated 1h (R/W) = Interrupt was generated

### 15.12.2.18 QCLR Register (Offset = 1Ah) [Reset = 0000h]

QCLR is shown in [Figure 15-43](#) and described in [Table 15-24](#).

Return to the [Summary Table](#).

QEP Interrupt Clear

**Figure 15-43. QCLR Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0/W1S-0h							

**Table 15-24. QCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R-0/W1S	0h	Clear QMA Error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
11	UTO	R-0/W1S	0h	Clear unit time out interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
10	IEL	R-0/W1S	0h	Clear index event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
9	SEL	R-0/W1S	0h	Clear strobe event latch interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
8	PCM	R-0/W1S	0h	Clear eQEP compare match event interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
7	PCR	R-0/W1S	0h	Clear position-compare ready interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
6	PCO	R-0/W1S	0h	Clear position counter overflow interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
5	PCU	R-0/W1S	0h	Clear position counter underflow interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
4	WTO	R-0/W1S	0h	Clear watchdog timeout interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag

**Table 15-24. QCLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	QDC	R-0/W1S	0h	Clear quadrature direction change interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
2	PHE	R-0/W1S	0h	Clear quadrature phase error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
1	PCE	R-0/W1S	0h	Clear position counter error interrupt flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag
0	INT	R-0/W1S	0h	Global interrupt clear flag Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Clears the interrupt flag

### 15.12.2.19 QFRC Register (Offset = 1Bh) [Reset = 0000h]

QFRC is shown in [Figure 15-44](#) and described in [Table 15-25](#).

Return to the [Summary Table](#).

QEP Interrupt Force

**Figure 15-44. QFRC Register**

15	14	13	12	11	10	9	8
RESERVED			QMAE	UTO	IEL	SEL	PCM
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 15-25. QFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12	QMAE	R/W	0h	Force QMA error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
11	UTO	R/W	0h	Force unit time out interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
10	IEL	R/W	0h	Force index event latch interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
9	SEL	R/W	0h	Force strobe event latch interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
8	PCM	R/W	0h	Force position-compare match interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
7	PCR	R/W	0h	Force position-compare ready interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
6	PCO	R/W	0h	Force position counter overflow interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
5	PCU	R/W	0h	Force position counter underflow interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
4	WTO	R/W	0h	Force watchdog time out interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt

**Table 15-25. QFRC Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	QDC	R/W	0h	Force quadrature direction change interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
2	PHE	R/W	0h	Force quadrature phase error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
1	PCE	R/W	0h	Force position counter error interrupt Reset type: SYSRSn 0h (R/W) = No effect 1h (R/W) = Force the interrupt
0	RESERVED	R	0h	Reserved

### 15.12.2.20 QEPSTS Register (Offset = 1Ch) [Reset = 0000h]

QEPSTS is shown in Figure 15-45 and described in Table 15-26.

Return to the [Summary Table](#).

QEP Status

**Figure 15-45. 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/W1C-0h	R-0h	R-0h	R-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R-0h

**Table 15-26. QEPSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	UPEVNT	R/W1C	0h	Unit position event flag Reset type: SYSRSn 0h (R/W) = No unit position event detected 1h (R/W) = Unit position event detected. Write 1 to clear
6	FIDF	R	0h	Direction on the first index marker Status of the direction is latched on the first index event marker. Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) on the first index event 1h (R/W) = Clockwise rotation (or forward movement) on the first index event
5	QDF	R	0h	Quadrature direction flag Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) 1h (R/W) = Clockwise rotation (or forward movement)
4	QDLF	R	0h	eQEP direction latch flag Reset type: SYSRSn 0h (R/W) = Counter-clockwise rotation (or reverse movement) on index event marker 1h (R/W) = Clockwise rotation (or forward movement) on index event marker
3	COEF	R/W1C	0h	Capture overflow error flag Reset type: SYSRSn 0h (R/W) = Overflow has not occurred. 1h (R/W) = Overflow occurred in eQEP Capture timer (QEPCTMR). This bit is cleared by writing a '1'.
2	CDEF	R/W1C	0h	Capture direction error flag Reset type: SYSRSn 0h (R/W) = Capture direction error has not occurred. 1h (R/W) = Direction change occurred between the capture position event. This bit is cleared by writing a '1'.
1	FIMF	R/W1C	0h	First index marker flag Reset type: SYSRSn 0h (R/W) = First index pulse has not occurred. 1h (R/W) = Set by first occurrence of index pulse. This bit is cleared by writing a '1'.

**Table 15-26. QEPSTS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	PCEF	R	0h	Position counter error flag. This bit is not sticky and it is updated for every index event. Reset type: SYSRSn 0h (R/W) = No error occurred during the last index transition 1h (R/W) = Position counter error

**15.12.2.21 QCTMR Register (Offset = 1Dh) [Reset = 0000h]**

QCTMR is shown in [Figure 15-46](#) and described in [Table 15-27](#).

Return to the [Summary Table](#).

QEP Capture Timer

**Figure 15-46. QCTMR Register**

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

**Table 15-27. QCTMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCTMR	R/W	0h	This register provides time base for edge capture unit. Reset type: SYSRSn

### 15.12.2.22 QCPRD Register (Offset = 1Eh) [Reset = 0000h]

QCPRD is shown in [Figure 15-47](#) and described in [Table 15-28](#).

Return to the [Summary Table](#).

QEP Capture Period

**Figure 15-47. QCPRD Register**

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

**Table 15-28. QCPRD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCPRD	R/W	0h	This register holds the period count value between the last successive eQEP position events Reset type: SYSRSn

### 15.12.2.23 QCTMRLAT Register (Offset = 1Fh) [Reset = 0000h]

QCTMRLAT is shown in [Figure 15-48](#) and described in [Table 15-29](#).

Return to the [Summary Table](#).

QEP Capture Latch

**Figure 15-48. QCTMRLAT Register**

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

**Table 15-29. QCTMRLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCTMRLAT	R	0h	The eQEP capture timer value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. Reset type: SYSRSn

### 15.12.2.24 QCPRDLAT Register (Offset = 20h) [Reset = 0000h]

QCPRDLAT is shown in [Figure 15-49](#) and described in [Table 15-30](#).

Return to the [Summary Table](#).

QEP Capture Period Latch

**Figure 15-49. QCPRDLAT Register**

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

**Table 15-30. QCPRDLAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	QCPRDLAT	R	0h	eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. Reset type: SYSRSn

**15.12.2.25 REV Register (Offset = 30h) [Reset = 0000009h]**

REV is shown in [Figure 15-50](#) and described in [Table 15-31](#).

Return to the [Summary Table](#).

QEP Revision Number

**Figure 15-50. REV Register**

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

**Table 15-31. REV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R-0	0h	Reserved
5-3	MINOR	R	1h	This field specifies the Minor Revision number for the eQEP IP. Reset type: N/A
2-0	MAJOR	R	1h	This field specifies the Major Revision number for the eQEP IP. Reset type: N/A

### 15.12.2.26 QEPSTROBESEL Register (Offset = 32h) [Reset = 0000000h]

QEPSTROBESEL is shown in [Figure 15-51](#) and described in [Table 15-32](#).

Return to the [Summary Table](#).

QEP Strobe select register

**Figure 15-51. QEPSTROBESEL Register**

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

**Table 15-32. QEPSTROBESEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R-0	0h	Reserved
1-0	STROBESEL	R/W	0h	Strobe source select: Reset type: SYSRSn 0h (R/W) = QEP Strobe after polarity mux 1h (R/W) = QEP Strobe after polarity mux 2h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCA 3h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCA

**15.12.2.27 QMACTRL Register (Offset = 34h) [Reset = 0000000h]**

QMACTRL is shown in [Figure 15-52](#) and described in [Table 15-33](#).

Return to the [Summary Table](#).

QMA Control register

**Figure 15-52. QMACTRL Register**

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

**Table 15-33. QMACTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R-0	0h	Reserved
2-0	MODE	R/W	0h	Select Mode for QMA mode: 000 : QMA Module is bypassed. 001 : QMA Mode-1 operation selected 010 : QMA Mode-2 operation selected 011 : QMA Module is bypassed (reserved) 1xx : QMA Module is bypassed (reserved) Reset type: SYSRSn

**15.12.2.28 QEPSRCSEL Register (Offset = 36h) [Reset = 0000000h]**

 QEPSRCSEL is shown in [Figure 15-53](#) and described in [Table 15-34](#).

 Return to the [Summary Table](#).

QEP Source Select Register

**Figure 15-53. QEPSRCSEL 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
QEPSSEL				QEPISEL				QEPBSEL				QEPASEL			
R/W-0h				R/W-0h				R/W-0h				R/W-0h			

**Table 15-34. QEPSRCSEL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-12	QEPSSEL	R/W	0h	QEP Strobe source select: 0x0: From device Pins (Default). Others: Tied to zero. Reset type: SYSRSn
11-8	QEPISEL	R/W	0h	QEP Index source select: 0x0: Device Pin (Default) 0x1: CMPSS1.CTRIPH 0x2: CMPSS2_LITE.CTRIPH 0x3: CMPSS3_LITE.CTRIPH 0x4: CMPSS4_LITE.CTRIPH 0x5: RSVD 0x6: RSVD 0x7: RSVD 0x8: RSVD 0x9: PWMXBAR.1 0xA: PWMXBAR.2 0xB: PWMXBAR.3 0xC: PWMXBAR.4 0xD: PWMXBAR.5 0xE: PWMXBAR.6 0xF: PWMXBAR.7 Note: eQEP needs to be disabled before configuring these bits as it can lead to unexpected behavior if eQEP is running. Reset type: SYSRSn

**Table 15-34. QEPRSEL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-4	QEPBSEL	R/W	0h	QEPB source select: 0x0: Device Pin (Default) 0x1: CMPSS1.CTRIPH 0x2: CMPSS2_LITE.CTRIPH 0x3: CMPSS3_LITE.CTRIPH 0x4: CMPSS4_LITE.CTRIPH 0x5: RSVD 0x6: RSVD 0x7: RSVD 0x8: RSVD 0x9: PWMXBAR.1 0xA:PWMXBAR.2 0xB:PWMXBAR.3 0xC:PWMXBAR.4 0xD:PWMXBAR.5 0xE:PWMXBAR.6 0xF:PWMXBAR.7 Note: eQEP needs to be disabled before configuring these bits as it can lead to unexpected behavior if eQEP is running. Reset type: SYSRSn
3-0	QEPASEL	R/W	0h	QEPA source select: 0x0: Device Pin (Default) 0x1: CMPSS1.CTRIPH 0x2: CMPSS2_LITE.CTRIPH 0x3: CMPSS3_LITE.CTRIPH 0x4: CMPSS4_LITE.CTRIPH 0x5: RSVD 0x6: RSVD 0x7: RSVD 0x8: RSVD 0x9: PWMXBAR.1 0xA:PWMXBAR.2 0xB:PWMXBAR.3 0xC:PWMXBAR.4 0xD:PWMXBAR.5 0xE:PWMXBAR.6 0xF:PWMXBAR.7 Note: eQEP needs to be disabled before configuring these bits as it can lead to unexpected behavior if eQEP is running. Reset type: SYSRSn

### 15.12.3 EQEP Registers to Driverlib Functions

**Table 15-35. EQEP Registers to Driverlib Functions**

File	Driverlib Function
<b>QPOSCNT</b>	
eqep.h	EQEP_getPosition
eqep.h	EQEP_setPosition
<b>QPOSINIT</b>	
eqep.h	EQEP_setInitialPosition
<b>QPOSMAX</b>	
eqep.h	EQEP_setPositionCounterConfig
<b>QPOSCMP</b>	
eqep.c	EQEP_setCompareConfig
<b>QPOSILAT</b>	
eqep.h	EQEP_getIndexPositionLatch
<b>QPOSSLAT</b>	

**Table 15-35. EQEP Registers to Driverlib Functions (continued)**

File	Driverlib Function
eqep.h	EQEP_getStrobePositionLatch
<b>QOSLAT</b>	
eqep.h	EQEP_getPositionLatch
<b>QUTMR</b>	
-	
<b>QUPRD</b>	
eqep.h	EQEP_loadUnitTimer
eqep.h	EQEP_enableUnitTimer
<b>QWDTMR</b>	
eqep.h	EQEP_setWatchdogTimerValue
eqep.h	EQEP_getWatchdogTimerValue
<b>QWDPRD</b>	
eqep.h	EQEP_enableWatchdog
<b>QDECCTL</b>	
eqep.c	EQEP_setCompareConfig
eqep.c	EQEP_setInputPolarity
eqep.h	EQEP_setDecoderConfig
eqep.h	EQEP_enableDirectionChangeDuringIndex
eqep.h	EQEP_disableDirectionChangeDuringIndex
<b>QEPCTL</b>	
eqep.h	EQEP_enableModule
eqep.h	EQEP_disableModule
eqep.h	EQEP_setPositionCounterConfig
eqep.h	EQEP_enableUnitTimer
eqep.h	EQEP_disableUnitTimer
eqep.h	EQEP_enableWatchdog
eqep.h	EQEP_disableWatchdog
eqep.h	EQEP_setPositionInitMode
eqep.h	EQEP_setSWPositionInit
eqep.h	EQEP_setLatchMode
eqep.h	EQEP_setEmulationMode
<b>QCAPCTL</b>	
eqep.h	EQEP_setCaptureConfig
eqep.h	EQEP_enableCapture
eqep.h	EQEP_disableCapture
<b>QPOSCTL</b>	
eqep.c	EQEP_setCompareConfig
eqep.h	EQEP_enableCompare
eqep.h	EQEP_disableCompare
eqep.h	EQEP_setComparePulseWidth
<b>QEINT</b>	
eqep.h	EQEP_enableInterrupt
eqep.h	EQEP_disableInterrupt
<b>QFLG</b>	
eqep.h	EQEP_getInterruptStatus

**Table 15-35. EQEP Registers to Driverlib Functions (continued)**

File	Driverlib Function
eqep.h	EQEP_getError
<b>QCLR</b>	
eqep.h	EQEP_clearInterruptStatus
<b>QFRC</b>	
eqep.h	EQEP_forceInterrupt
<b>QEPSTS</b>	
eqep.h	EQEP_getDirection
eqep.h	EQEP_getStatus
eqep.h	EQEP_clearStatus
<b>QCTMR</b>	
eqep.h	EQEP_getCaptureTimer
eqep.h	EQEP_getCaptureTimerLatch
<b>QCPRD</b>	
eqep.h	EQEP_getCapturePeriod
eqep.h	EQEP_getCapturePeriodLatch
<b>QCTMRLAT</b>	
eqep.h	EQEP_getCaptureTimerLatch
<b>QCPRDLAT</b>	
eqep.h	EQEP_getCapturePeriodLatch
<b>REV</b>	
-	
<b>QEPSTROBESEL</b>	
eqep.h	EQEP_setStrobeSource
<b>QMACTRL</b>	
eqep.h	EQEP_setQMAModuleMode
<b>QEPSRCSEL</b>	
eqep.h	EQEP_selectSource

This page intentionally left blank.

## Chapter 16 Controller Area Network (CAN)



This chapter describes the Controller Area Network (CAN) module. CAN is a serial communications protocol that efficiently supports distributed real-time control with a high level of reliability. The CAN module supports bit rates up to 1 Mbit/s and is compliant with the ISO11898-1 (CAN 2.0B) protocol specification.

Further information can be found in:

- [Calculator for CAN Bit Timing Parameters Application Report](#)
- [Programming Examples and Debug Strategies for the DCAN Module Application Report](#)
- [Configurable Error Generator for Controller Area Network Application Report](#)

<b>16.1 Introduction</b> .....	<b>1632</b>
<b>16.2 Functional Description</b> .....	<b>1635</b>
<b>16.3 Operating Modes</b> .....	<b>1637</b>
<b>16.4 Multiple Clock Source</b> .....	<b>1643</b>
<b>16.5 Interrupt Functionality</b> .....	<b>1644</b>
<b>16.6 Parity Check Mechanism</b> .....	<b>1646</b>
<b>16.7 Debug Mode</b> .....	<b>1647</b>
<b>16.8 Module Initialization</b> .....	<b>1647</b>
<b>16.9 Configuration of Message Objects</b> .....	<b>1648</b>
<b>16.10 Message Handling</b> .....	<b>1649</b>
<b>16.11 CAN Bit Timing</b> .....	<b>1655</b>
<b>16.12 Message Interface Register Sets</b> .....	<b>1664</b>
<b>16.13 Message RAM</b> .....	<b>1666</b>
<b>16.14 Software</b> .....	<b>1671</b>
<b>16.15 CAN Registers</b> .....	<b>1673</b>

## 16.1 Introduction

This device uses the CAN IP known as DCAN.

### 16.1.1 DCAN Related Collateral

#### Foundational Materials

- [Automotive CAN Overview and Training](#) (Video)
- [CAN Physical layer](#) (Video)
- [CAN and CAN FD Overview](#) (Video)
- [CAN and CAN FD Protocol](#) (Video)

### 16.1.2 Features

The CAN module implements the following features:

- Complies with ISO11898-1 (Bosch® CAN protocol specification 2.0 A and B)
- Bit rates up to 1Mbps
- Multiple clock sources
- 32 message objects (“message objects” are also referred to as “mailboxes” in this document; the two terms are used interchangeably), each with the following properties:
  - Configurable as receive or transmit
  - Configurable with standard (11-bit) or extended (29-bit) identifier
  - Supports programmable identifier receive mask
  - Supports data and remote frames
  - Holds 0 to 8 bytes of data
  - Parity-checked configuration and data RAM
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus-on, after bus-off state by a programmable 32-bit timer
- Message-RAM parity-check mechanism
- Two interrupt lines

---

#### Note

For a CAN bit clock of 100MHz, the smallest bit rate possible is 3.90625kbps.

Depending on the timing settings used, the accuracy of the on-chip zero-pin oscillator (specified in the data sheet) may not meet the requirements of the CAN protocol. In this situation, an external clock source must be used.

---

### 16.1.3 Block Diagram

Figure 16-1 shows a block diagram of the CAN module. Following is a description of some of the main blocks.

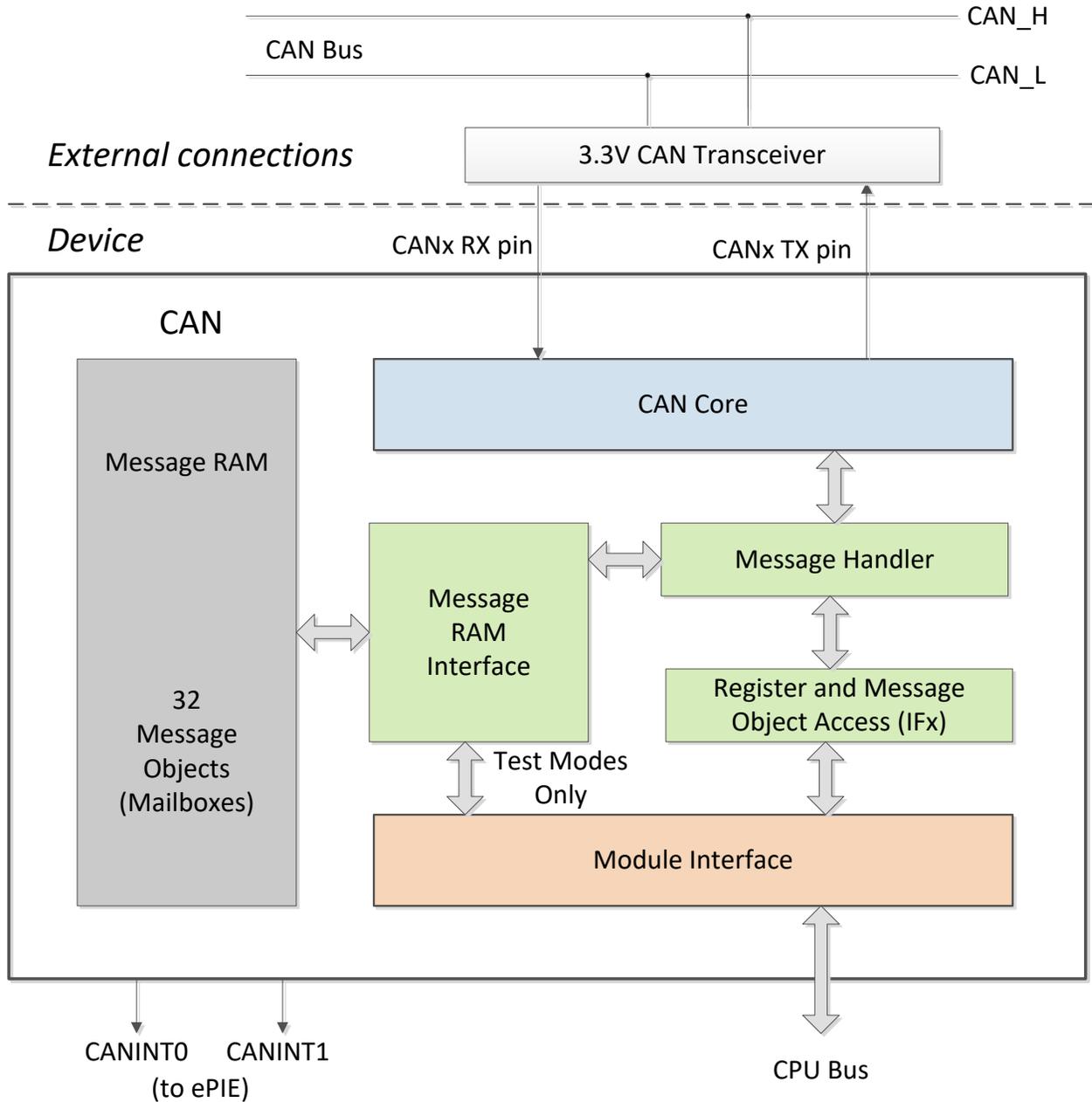


Figure 16-1. CAN Block Diagram

### 16.1.3.1 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.

### 16.1.3.2 Message Handler

The message handler is a state machine which controls the data transfer between the single-port Message RAM and the CAN Core's Rx/Tx Shift register. It also handles acceptance filtering and the interrupt request generation as programmed in the control registers.

### 16.1.3.3 Message RAM

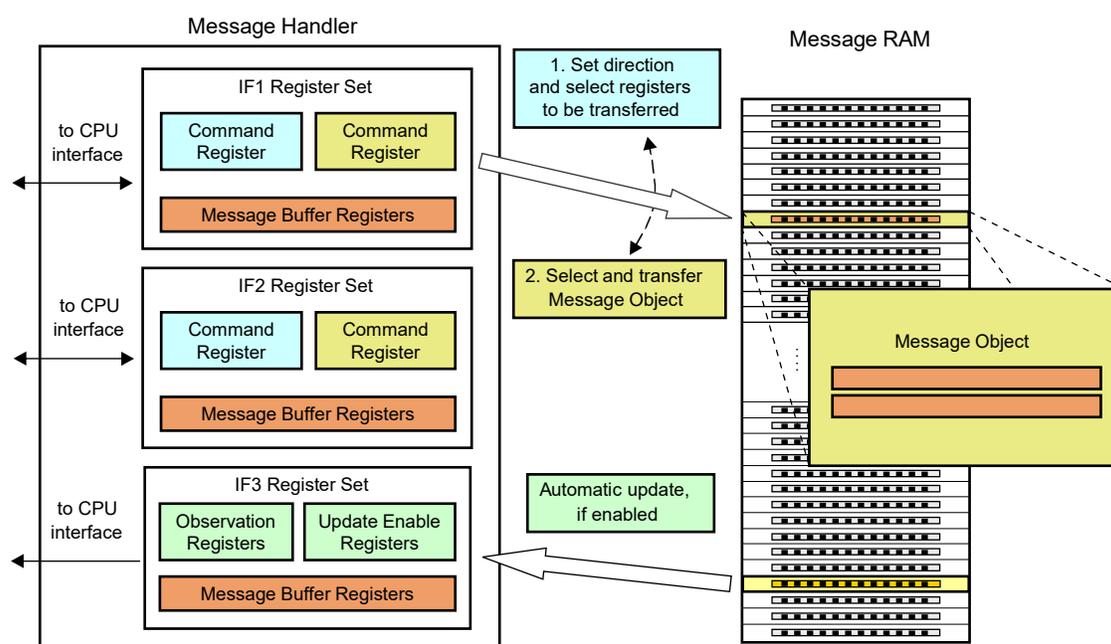
The CAN message RAM enables the storage of 32 CAN messages.

### 16.1.3.4 Registers and Message Object Access (IFx)

Data consistency is provided by indirect accesses to the message objects. During normal operation, all CPU accesses to the message RAM are done through Interface registers. The IFx registers can be thought of as a "window" through which the message objects (mailboxes) are accessed.

Three Interface register sets control the CPU read and write accesses to the Message RAM, see [Figure 16-2](#). 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 16.12](#). 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.



**Figure 16-2. Accessing Message Objects Through IFx Registers**

## 16.2 Functional Description

The CAN module performs CAN protocol communication according to ISO 11898-1. The bit rate can be programmed to values up to 1Mbps. A CAN transceiver chip is required for the connection to the physical layer (CAN bus).

For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the message handler. These functions are: acceptance filtering; the transfer of messages between the CAN Core and the Message RAM; and the handling of transmission requests .

The register set of the CAN can be accessed directly by the CPU through the module interface. These registers are used to control and configure the CAN core and the message handler, and to access the message RAM.

### 16.2.1 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some I/O functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification must be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pull-ups can be configured in the GPyPUD register.

See the *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux and settings.

### 16.2.2 Address/Data Bus Bridge

The CAN module uses a special addressing scheme to support byte accesses. It is recommended to only use 32-bit accesses to the CAN registers using the *HWREG\_BP()* macro that uses the *\_\_byte\_peripheral\_32()* intrinsic. If 16-bit accesses are to be used, the lower 16-bits must be written to the register's address, and the upper 16-bits must be written to the register's address plus 2.

Because of the bus bridge, the view of the CAN module's register space through the Code Composer Studio™ (CCS) IDE memory window does not always match the actual addressing. When the view mode is 32-bit or 16-bit, even addresses are effectively duplicated; odd addresses can be ignored. When the view mode is 8-bit, even addresses from within the CAN module are duplicated into the odd addresses in the CCS memory view; odd addresses from the module are not displayed.

**Table 16-1. CAN Register Access from Software**

CAN Register Space			C28x 8-Bit		C28x 16-Bit		C28x 32-Bit	
Address	Name	Data	Access	Data	Access	Data	Address	Data
0x00	CAN_CTL	0x33221100	__byte((int *)0x00,0)	0x0000	*((short *)0x00))	0x1100	*((long *)0x00))	0x33221100
0x04	CAN_ES	0x77665544	__byte((int *)0x01,0)	0x0011	*((short *)0x01))	0x1100	*((long *)0x01))	0x33221100
0x08	CAN_ERRC	0xBBA9988	__byte((int *)0x02,0)	0x0022	*((short *)0x02))	0x3322	*((long *)0x02))	0x33221100
0x0C	CAN_BTR	0xFFEEDDCC	__byte((int *)0x03,0)	0x0033	*((short *)0x03))	0x3322	*((long *)0x03))	0x33221100
			__byte((int *)0x04,0)	0x0044	*((short *)0x04))	0x5544	*((long *)0x04))	0x77665544
			__byte((int *)0x05,0)	0x0055	*((short *)0x05))	0x5544	*((long *)0x05))	0x77665544
			__byte((int *)0x06,0)	0x0066	*((short *)0x06))	0x7766	*((long *)0x06))	0x77665544
			__byte((int *)0x07,0)	0x0077	*((short *)0x07))	0x7766	*((long *)0x07))	0x77665544
			__byte((int *)0x08,0)	0x0088	*((short *)0x08))	0x9988	*((long *)0x08))	0xBBA9988
			__byte((int *)0x09,0)	0x0099	*((short *)0x09))	0x9988	*((long *)0x09))	0xBBA9988
			__byte((int *)0x0A,0)	0x00AA	*((short *)0x0A))	0xBBAA	*((long *)0x0A))	0xBBA9988
			__byte((int *)0x0B,0)	0x00BB	*((short *)0x0B))	0xBBAA	*((long *)0x0B))	0xBBA9988
			__byte((int *)0x0C,0)	0x00CC	*((short *)0x0C))	0xDDCC	*((long *)0x0C))	0xFFEEDDCC
			__byte((int *)0x0D,0)	0x00DD	*((short *)0x0D))	0xDDCC	*((long *)0x0D))	0xFFEEDDCC
			__byte((int *)0x0E,0)	0x00EE	*((short *)0x0E))	0xFFEE	*((long *)0x0E))	0xFFEEDDCC
			__byte((int *)0x0F,0)	0x00FF	*((short *)0x0F))	0xFFEE	*((long *)0x0F))	0xFFEEDDCC

**Table 16-2. CAN Register Access from Code Composer Studio™ IDE**

CCS 8-Bit		CCS 16-Bit		CCS 32-Bit	
Address	Displayed Data	Address	Displayed Data	Address	Displayed Data
0x00	0x00	0x00	0x1100	0x00	0x11001100
0x01	0x00	0x01	0x1100	0x02	0x33223322
0x02	0x22	0x02	0x3322	0x04	0x55445544
0x03	0x22	0x03	0x3322	0x06	0x77667766
0x04	0x44	0x04	0x5544	0x08	0x99889988
0x05	0x44	0x05	0x5544	0x0A	0xBBAABBAA
0x06	0x66	0x06	0x7766	0x0C	0xDDCCDDCC
0x07	0x66	0x07	0x7766	0x0E	0xFFEEFFEE
0x08	0x88	0x08	0x9988		
0x09	0x88	0x09	0x9988		
0x0A	0xAA	0x0A	0xBBAA		
0x0B	0xAA	0x0B	0xBBAA		
0x0C	0xCC	0x0C	0xDDCC		
0x0D	0xCC	0x0D	0xDDCC		
0x0E	0xEE	0x0E	0xFFEE		
0x0F	0xEE	0x0F	0xFFEE		

## 16.3 Operating Modes

### 16.3.1 Initialization

The initialization mode is entered either by software (by setting the **Init** bit in the CAN\_CTL register), by hardware reset, or by going bus-off. While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN\_TX output is recessive (high). The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

To initialize the CAN controller, the CPU has to configure the CAN bit timing and those message objects that are used for CAN communication. Message objects that are not needed, can be deactivated with the MsgVal bits cleared.

The access to the Bit Timing Register for the configuration of the bit timing is enabled when both **Init** and CCE bits in the CAN Control register are set.

Clearing the Init bit finishes the software initialization. Afterwards, the bit stream processor (BSP) synchronizes to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before the BSP can take part in bus activities and start the message transfer. For more details, see [Section 16.11](#).

The initialization of the message objects is independent of the Init bit; however, all message objects must be configured with 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 is applied to it when the modified message object number is same or smaller than the previously found message object. This makes sure of data consistency even when changing message objects; for example, while there is a pending CAN frame reception.

### 16.3.2 CAN Message Transfer (Normal Operation)

Once the CAN is initialized and the Init bit is reset to zero, the CAN Core synchronizes to the CAN bus and is ready for communication.

Received messages are stored into the appropriate message objects, if the messages pass acceptance filtering. The whole message (MSGID, DLC, and up to 8 data bytes) is stored into the message object. As a consequence, for example, if the identifier mask is used, the MSGID bits that are masked to "don't care" can change in the message object when a received message is stored.

The CPU can read or write each message at any time using the interface registers, as the message handler provides data consistency in case of concurrent accesses.

Messages to be transmitted can be updated by the CPU. If a permanent message object (MSGID, control bits set up during configuration, and setup for multiple CAN transfers) exists for the message, it is possible to only update the data bytes. If several transmit messages must 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 can be requested at the same time. The message objects are subsequently transmitted, according to the internal priority. Messages can be updated or set to "not valid" at any time, even if a requested transmission is still pending. However, the data bytes are discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission can be automatically requested by the reception of a remote frame with a matching identifier.

#### 16.3.2.1 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides a mechanism to automatically retransmit frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service is not confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled and can be disabled by setting the DAR bit in the CAN control register. Further details to this mode are provided in [Section 16.10.3](#).

#### 16.3.2.2 Auto-Bus-On

After the CAN has entered the bus-off state, the CPU can start a bus-off-recovery sequence by resetting the *Init* bit. If this is not done, the module stays in the bus-off state.

The CAN provides an automatic auto-bus-on feature that is enabled by the ABO bit. If set, the CAN automatically starts the bus-off-recovery sequence. The sequence can be delayed by a user-defined number of clock cycles.

---

#### Note

If the CAN module goes Bus-Off due to multiple CAN bus errors, the CAN module stops all bus activities and automatically sets the Init bit. Once the Init bit is cleared by the application (or due to the auto-bus-on feature), the device waits for 129 occurrences of Bus Idle (equal to  $129 * 11$  consecutive recessive bits) before resuming normal operation. The Bus-Off recovery sequence cannot be shortened by setting or resetting the Init bit. At the end of the bus-off recovery sequence, the error counters reset. After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register. This enables the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

---

### 16.3.3 Test Modes

The CAN module provides several test modes that are mainly intended for self-test purposes. Figure 16-3 aids in understanding the various test modes. Figure 16-3 must be viewed as representative of the module behavior, and not as a gate-accurate implementation of the module. Figure 16-3 does not include the GPIO muxing or the I/O buffers.

For all test modes, the Test bit in the CAN control register needs to be set to 1 to enable write access to the CAN\_TEST register.

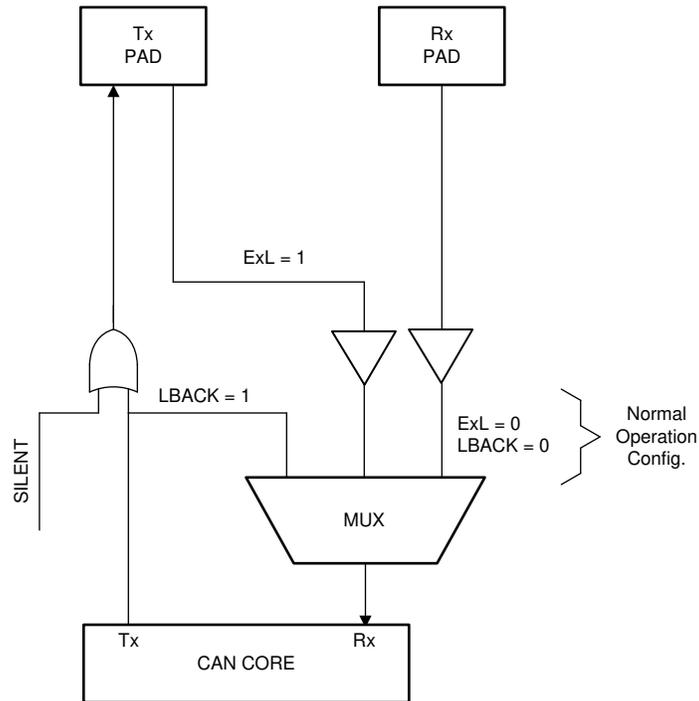


Figure 16-3. CAN\_MUX

### 16.3.3.1 Silent Mode

The silent mode can be used to analyze the traffic on the CAN bus without affecting the CAN 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 the CAN does not send any dominant bits. However, the received frames are internally routed to the CAN Core.

Figure 16-4 shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in silent mode. Silent mode can be activated by setting the Silent bit in test register (CAN\_TEST), to 1. In ISO 11898-1, the silent mode is called the bus monitoring mode.

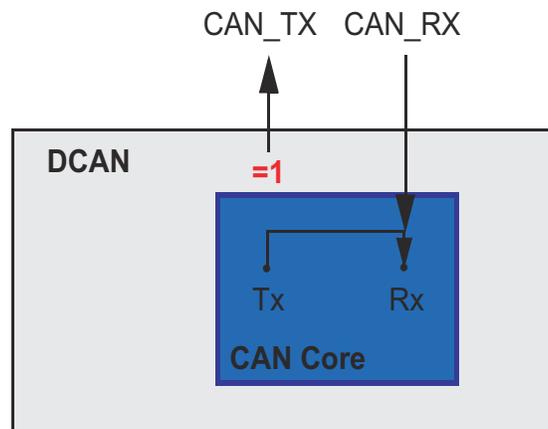


Figure 16-4. CAN Core in Silent Mode

### 16.3.3.2 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 the messages 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.

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 16-5 shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in loopback mode. Loopback mode can be activated by setting the LBack bit in the CAN\_TEST register to 1.

**Note**

In loopback mode, the signal path from the CAN core to the Tx pin, and the signal path from the Tx pin back to the CAN core are disregarded. For including these into the testing, see Section 16.3.3.3.

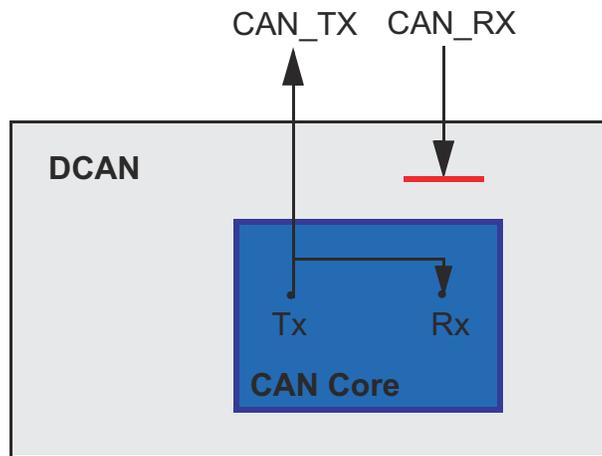


Figure 16-5. CAN Core in Loopback Mode

### 16.3.3.3 External Loopback Mode

The external loopback mode is similar to the loopback mode; however, the external loopback mode includes the signal path from the CAN core to the Tx pin, and the signal path from the Tx pin back to the CAN core. When the external loopback mode is selected, the CAN core is connected to the input buffer of the Tx pin. With this configuration, the Tx pin IO circuit can be tested. External loopback mode can be activated by setting the ExL bit in Test Register to 1.

Figure 16-6 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 is ignored.

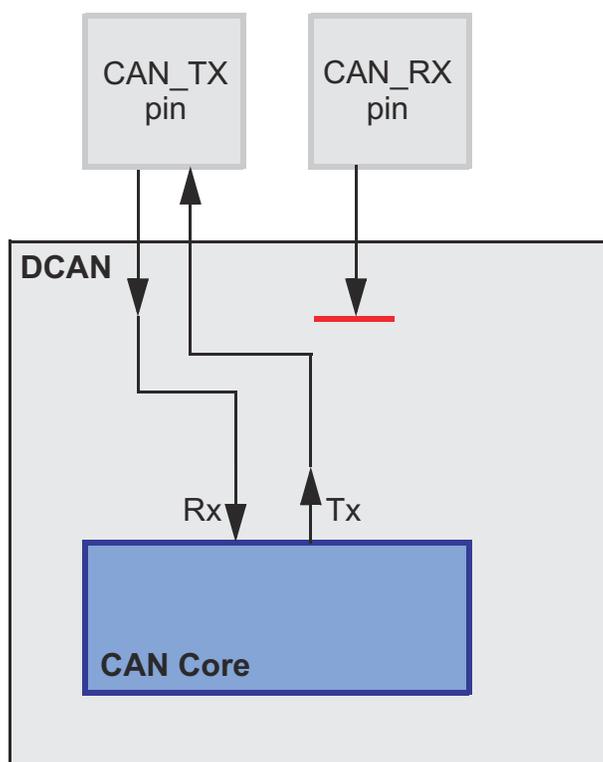
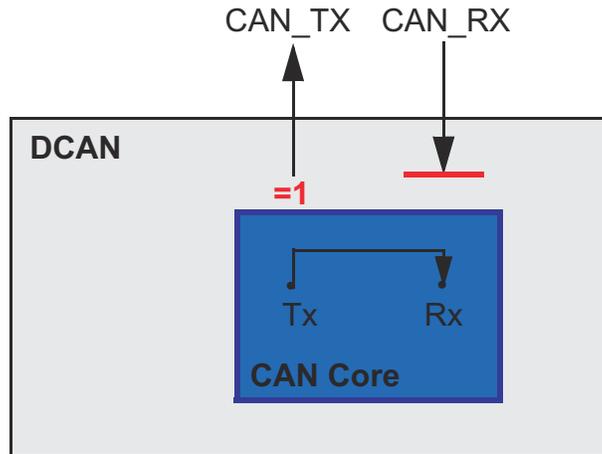


Figure 16-6. CAN Core in External Loopback Mode

### 16.3.3.4 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. 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 are sent on the CAN\_TX pin.

Figure 16-7 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 16-7. CAN Core in Loopback Combined with Silent Mode**

## 16.4 Multiple Clock Source

The CAN bit timing clock is normally derived from the system clock (SYSCLK). If desired, the bit clock can be derived directly from the external clock source (XTAL).

The *System Control and Interrupts* chapter and the device data sheet provide 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 8 clock cycles per bit time. To achieve a transfer rate of 1Mbps an oscillator frequency of 8MHz or higher has to be used.

## 16.5 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 and Int1ID in the Interrupt register. When no interrupt is pending, the register holds the value zero. Each interrupt line remains active until the dedicated field in the Interrupt register (Int0ID or Int1ID) again reaches zero (this means the cause of the interrupt is reset), or until IE0 and IE1 are reset. The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RxOk, TxOk, and LEC by reading the Error and Status Register, but a write access of the CPU never generates or resets 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 and INT1ID 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 that reads the message from the interrupt source can also read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1 or IF2 Command register). When IntPnd is cleared, the Interrupt register points to the next message object with a pending interrupt.

The CAN module features a module-level interrupt enable and acknowledge mechanism. To enable the CAN0 and CAN1 interrupts, you must set the appropriate bits in the CAN\_GLB\_INT\_EN register. When handling an interrupt, the individual message or status change flag must be cleared prior to acknowledging the interrupt using CAN\_GLB\_INT\_CLR and PIEACK.

### 16.5.1 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 16.13.1](#). Message object interrupts can be routed to the CAN0INT or CAN1INT line, which is controlled by the Interrupt Multiplexer register.

Note that writing to the IntPnd bit in the CAN\_IFnMCTL registers can force an interrupt.

### 16.5.2 Status Change Interrupts

The events RxOk, TxOk, and LEC in the Error and Status register belong to the status change interrupts. The status change interrupt group can be enabled by the SIE bit in the CAN Control Register. If SIE is set, a status change interrupt is generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration. Status Change interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN\_CTL Register.

### 16.5.3 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 the CAN\_CTL register.

### 16.5.4 Peripheral Interrupt Expansion (PIE) Module Nomenclature for DCAN Interrupts

[Table 16-3](#) shows the Peripheral Interrupt Expansion (PIE) module nomenclature for the interrupts.

**Table 16-3. PIE Module Nomenclature for Interrupts**

Interrupt	CANA	CANB
CANINT0	CANA_0	CANB_0
CANINT1	CANA_1	CANB_1

### 16.5.5 Interrupt Topologies

Interrupt topologies for CAN are illustrated in Figure 16-8 and Figure 16-9. Mailbox interrupts for transmit and receive operations can be routed to both CANINT0 and CANINT1. However, error and status interrupts can only be routed to CANINT0.

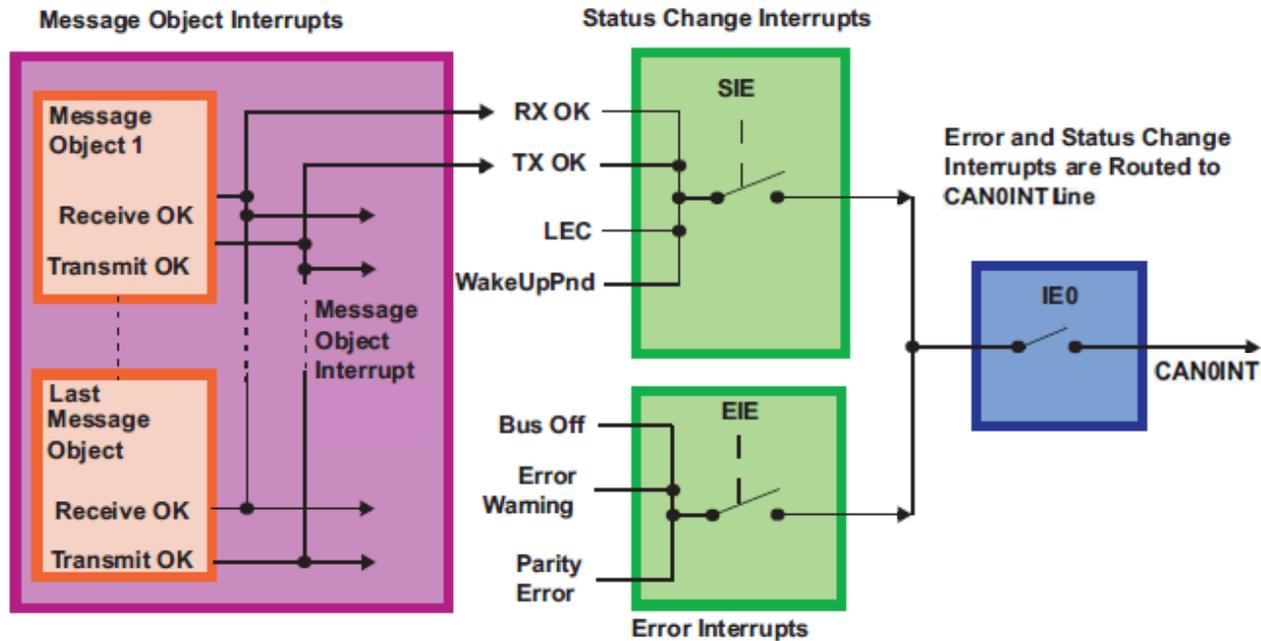


Figure 16-8. CAN Interrupt Topology 1

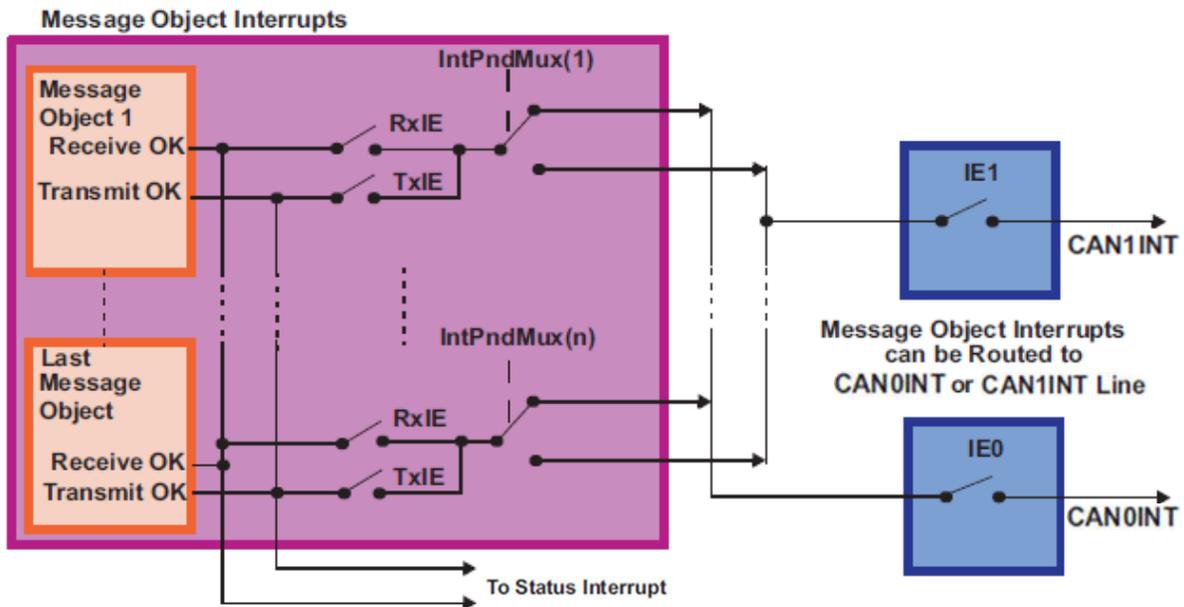


Figure 16-9. CAN Interrupt Topology 2

## 16.6 Parity Check Mechanism

The CAN provides a parity check mechanism to make sure data integrity of the message RAM data. For each word (32 bits) in Message RAM, one parity bit is calculated.

Parity information is stored in the Message RAM on write accesses and is checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by the PMD bit field in the CAN control register. In case of a disabled parity check, the parity bits in message RAM are left unchanged on write access to the data area and no check is done on read access.

If parity checking is enabled, parity bits are automatically generated and checked by the CAN. A parity bit is set if the modulo-2-sum of the data bits is 1. This means that if the parity bit is set, then there are an odd number of 1 bits in the data.

### 16.6.1 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 is checked. If a parity error is detected, the PER bit in the Error and Status register is set. If error interrupts are enabled, an interrupt can also be generated. To avoid the transmission of invalid data over the CAN bus, the MsgVal bit of the message object is reset.

The message object data can be read by the CPU, independently of parity errors. Thus, the application has to make sure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

## 16.7 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 waits until a transmission is started, a reception is finished, or the Bus idle state is recognized. If the IDS bit is set, the debugger immediately interrupts the current transmission or reception. Afterwards, the CAN enters Debug mode, indicated by the InitDbg flag, in the CAN Control register. During debug mode, all CAN registers can be accessed. Reading reserved bits returns a 0; writing to reserved bits has no effect. Also, the message RAM is memory-mapped, so this allows the external debug unit to read the message RAM. For the memory organization (see [Section 16.13.3](#)).

---

### Note

During debug mode, the Message RAM cannot be accessed using the IFx register sets.

Writing to control registers in Debug mode can 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

## 16.8 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 must be completed before the CAN protocol functions are enabled.

For the configuration of the message objects, see [Section 16.9](#).

For the configuration of the Bit Timing, see [Section 16.11.2](#).

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 using the CAN bus.

The CPU can enable the interrupt lines (setting IE0 and IE1 to 1) at the same time when the CPU clears Init and CCE. The status interrupts EIE and SIE can be enabled simultaneously.

The CAN communication can be controlled interrupt-driven or in polling mode. The Interrupt Register points to those message objects with IntPnd = 1. The register is updated even if the interrupt lines to the CPU are disabled (IE0 and IE1 are 0).

The CPU can 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.

## 16.9 Configuration of Message Objects

The entire Message RAM must be configured before the end of the initialization; however, it is also possible to change the configuration of message objects during CAN communication.

### 16.9.1 Configuration of a Transmit Object for Data Frames

Figure 16-10 shows how a transmit object can be initialized.

**Figure 16-10. 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. The arbitration bits define the identifier and type of the outgoing message. If an 11-bit Identifier (standard frame) is used (Xtd = 0), the Identifier 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 must not be set before the data is valid.
- If the TxIE bit is set, the IntPnd bit is set after a successful transmission of the message object.
- If the RmtEn bit is set, a matching received remote frame causes the TxRqst bit to be set; the remote frame is autonomously answered by a data frame.
- The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) can be used (UMask = 1) to allow groups of remote frames with similar identifiers to set the TxRqst bit. The Dir bit must not be masked. For details, see [Section 16.10.8](#). Identifier masking must be disabled (UMask = 0) if no remote frames are allowed to set the TxRqst bit (RmtEn = 0).

### 16.9.2 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 causes the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

### 16.9.3 Configuration of a Single Receive Object for Data Frames

Figure 16-11 shows how a receive object for data frames can be initialized.

**Figure 16-11. 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. The arbitration bits define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), the Identifier 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] is set to 0.
- When the message handler stores a data frame in the message object, the message handler stores the received data length code and the corresponding number of data bytes. If the data length code is less than 8, the remaining bytes of the message object can be overwritten by non-specified values.
- The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) can be used (UMask = 1) to allow groups of data frames with similar identifiers to be accepted. The Dir bit must 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 are overwritten by the bits of the stored data frame.
- If the RxIE bit is set, the IntPnd bit is 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 stored in the Arbitration bits is triggered. The content of the Arbitration bits can change if the Mask bits are used (UMask = 1) for acceptance filtering.

### 16.9.4 Configuration of a Single Receive Object for Remote Frames

Figure 16-12 shows how a receive object for remote frames can be initialized.

**Figure 16-12. 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 can be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object does not trigger the transmission of a data frame. Receive objects for remote frames can be expanded to a FIFO buffer, see [Section 16.9.5](#).
- UMask must be set to 1. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) can 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 must not be masked in typical applications. For details, see [Section 16.10.8](#).
- The arbitration bits (ID[28:0] and Xtd bit) can be given by the application. The arbitration bits 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 are overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = 0), the Identifier 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] is set to 0.
- The data length code (DLC[3:0]) can be given by the application. When the message handler stores a remote frame in the message object, the message handler stores the received data length code. The data bytes of the message object remain unchanged.
- If the RxIE bit is set, the IntPnd bit is set when a received remote frame is accepted and stored in the message object.

### 16.9.5 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 form a FIFO, the identifiers and masks (if used) of these message objects are programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number is the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one must be programmed to 0. The EoB bit of the last message object of a FIFO buffer is set to 1, configuring the last message object as the end of the block.

### 16.10 Message Handling

When initialization is finished, the CAN module synchronizes to the traffic on the CAN bus. The CAN module does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application must update the data of the messages to be transmitted and to enable and request the transmission. The transmission is requested automatically when a matching remote frame is received.

The application can read messages that are received and accepted. Messages that are not read before the next messages are accepted for the same message object are overwritten. Messages can be read interrupt-driven or after polling of NewDat.

### 16.10.1 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. The message handler state machine 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 using 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.

### 16.10.2 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, the requests are serviced according 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 can 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.

### 16.10.3 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 is reset. If TxIE is set, IntPnd is set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message is 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 are transmitted in the order of the 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 for storage. The remote frames automatically trigger the transmission of a data frame, if the *RmtEn* bit is set in the matching Transmit Object.

#### 16.10.4 Updating a Transmit Object

The CPU can update the data bytes of a transmit object any time using the IF1 and IF2 interface registers; neither MsgVal nor TxRqst need 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 must 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 can 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 16.10.3](#).

When NewDat is set together with TxRqst, NewDat is reset as soon as the new transmission has started.

#### 16.10.5 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 can 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 need to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, the transmission is continued; however, the transmission is not repeated if the transmission is disturbed.

To update only the data bytes of a message being transmitted, set bits [23:16] of the Command register to 0x87.

---

#### Note

After the update of the transmit object, the interface register set contains a copy of the actual contents of the object, including the part that had not been updated.

---

### 16.10.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the message handler starts to scan the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

### 16.10.7 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 makes sure 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 must reset the NewDat bit when the CPU reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

### 16.10.8 Reception of Remote Frames

When a remote frame is received, three different configurations of the matching message object are 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

### 16.10.9 Reading Received Messages

The CPU can read a received message any time using the IFx interface registers, the data consistency is provided by the message handler state machine.

Typically the CPU writes 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command Register. That combination transfers 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 the 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 is not automatically reset.

#### 16.10.10 Requesting New Data for a Receive Object

By means of a remote frame, the CPU can request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object causes 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 can 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.

#### 16.10.11 Storing Received Messages in FIFO Buffers

Several message objects can be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifiers. 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 bit 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 are written into the last message object of the FIFO Buffer (EoB = 1) and therefore overwrite previous messages in this message object.

#### 16.10.12 Reading from a FIFO Buffer

Several messages can be accumulated in a set of message objects that are concatenated to form a FIFO buffer before the application program is required (to avoid the loss of data) to empty the buffer. A FIFO buffer of length N stores N-1 plus the last received message since the last time the FIFO buffer was cleared. A FIFO buffer is cleared by reading and resetting the NewDat bits of all the message objects, starting at the FIFO object with the lowest message number. This can be done in a subroutine following the example shown in [Figure 16-13](#).

---

#### Note

All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise, true FIFO functionality cannot be maintained, since the message objects of a partly read buffer are refilled according to the normal (descending) priority.

---

Reading from a FIFO Buffer message object and resetting the NewDat bit is handled the same way as reading from a single message object.

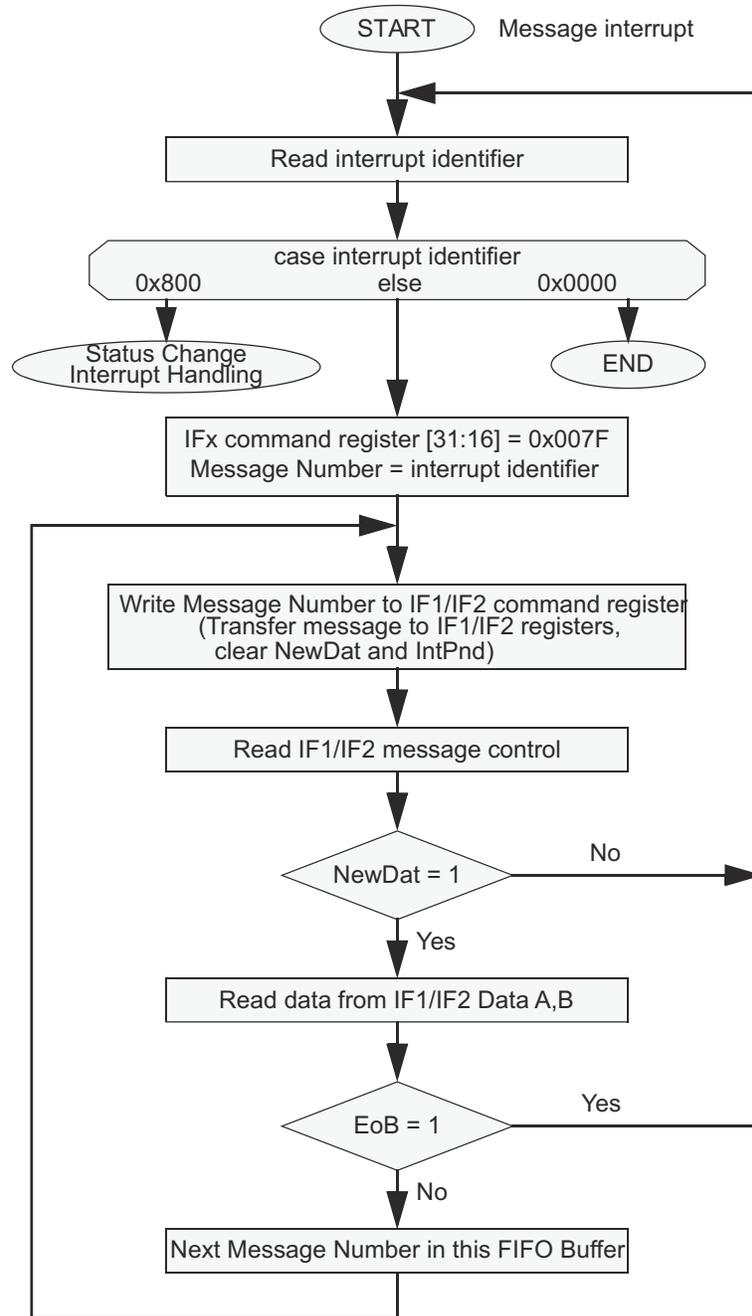


Figure 16-13. CPU Handling of a FIFO Buffer (Interrupt Driven)

## 16.11 CAN Bit Timing

The CAN supports bit rates up to 1000kBit/s.

Each CAN node has a 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 ( $F_{osc}$ ) can 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 amends 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, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point can cause one of the transmitters to become error passive.

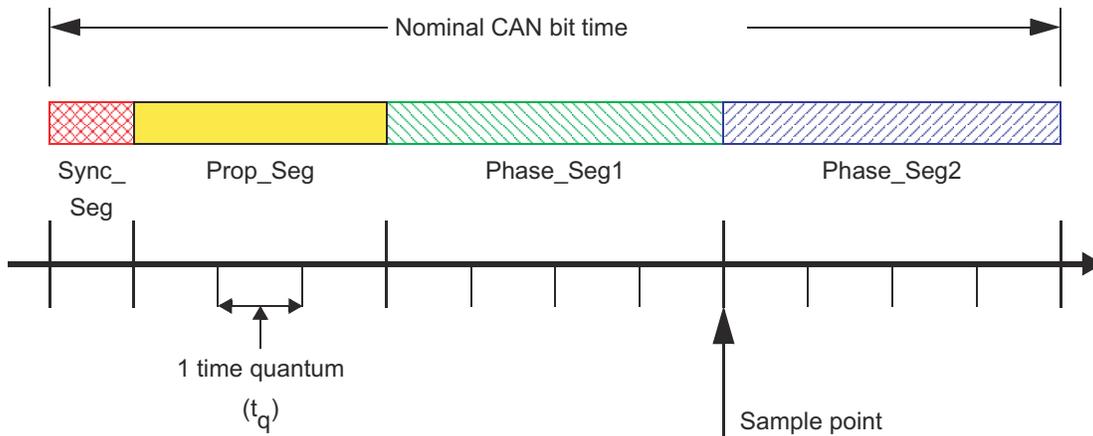
The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

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.

### 16.11.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see [Figure 16-14](#)):

- Synchronization Segment (Sync\_Seg)
- Propagation Time Segment (Prop\_Seg)
- Phase Buffer Segment 1 (Phase\_Seg1)
- Phase Buffer Segment 2 (Phase\_Seg2)



**Figure 16-14. 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 16-4](#) describes the minimum programmable ranges required by the CAN protocol.

A given bit rate can be met by different bit time configurations.

**Table 16-4. Programmable Ranges Required by CAN Protocol**

Parameter	Range	Remark
Sync_Seg	1 $t_q$ (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 ... 8] $t_q$	Compensates for the physical delay times
Phase_Seg1	[1 ... 8] $t_q$	Can be lengthened temporarily by synchronization
Phase_Seg2	[1 ... 8] $t_q$	Can be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 ... 4] $t_q$	Cannot 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 must be considered.

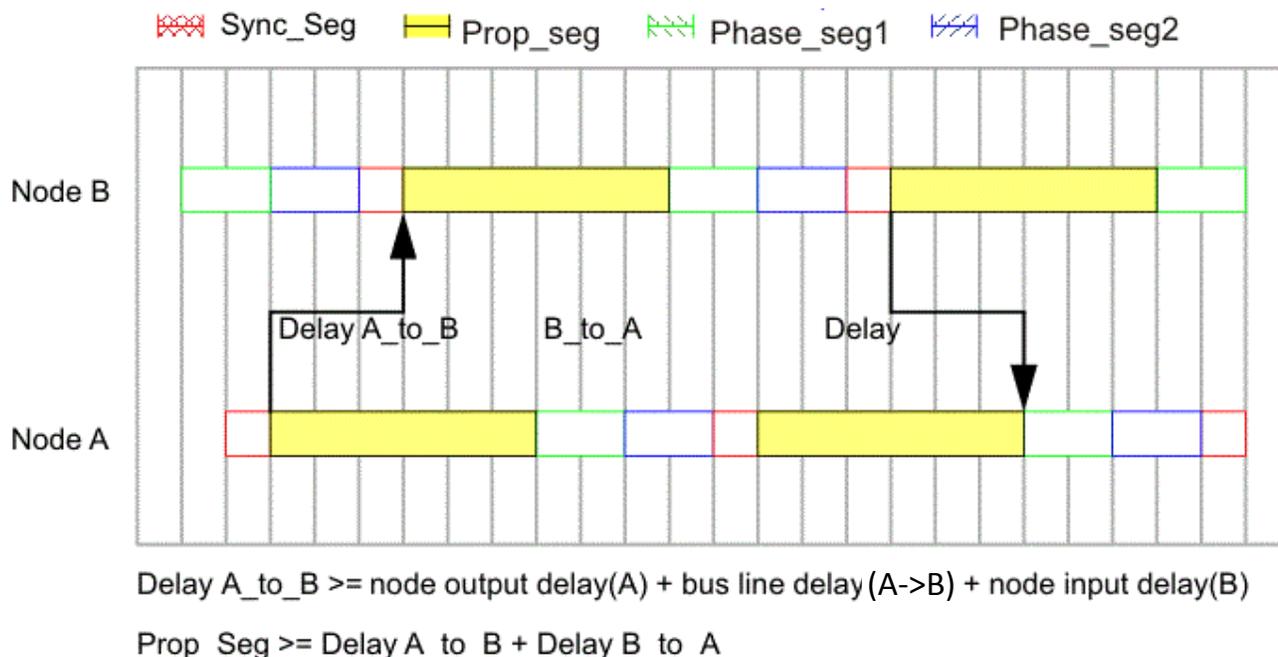
**16.11.1.1 Synchronization Segment**

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, the distance to the Sync\_Seg is called the phase error of this edge.

**16.11.1.2 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 16-15 shows the phase shift and propagation times between two CAN nodes.


**Figure 16-15. Propagation Time Segment**

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. Node A has sent a Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized to the received edge from recessive to dominant. Since node B has received this edge delay ( $A\_to\_B$ ) after the bit has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority, so node B wins the arbitration at a specific identifier bit when node B transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B arrives 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 arrives at node A after the start of Phase\_Seg1, node A can potentially sample a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus, which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the bit timing configuration (Prop\_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3-Sample Mode. The CAN module on this device does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of  $1 t_q$ , requiring a longer Prop\_Seg.

### 16.11.1.3 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 can be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism can 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. The 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 the sample with the bus level at the previous sample point. A synchronization can 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 the edge occurs inside of Sync\_Seg; otherwise, the 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, the phase error is positive.

Two types of synchronization exist: hard synchronization and resynchronization. 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 Resynchronization:** 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 that 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, Phase\_Seg1 is lengthened by SJW.

When the phase error of the edge that 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, Phase\_Seg2 is shortened by SJW.

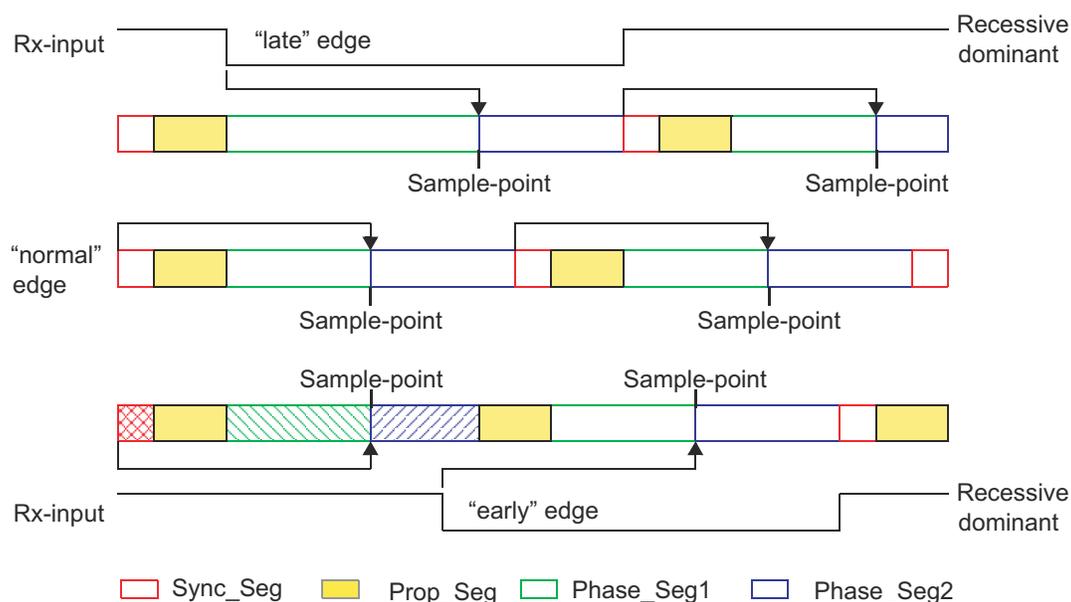
If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

Only one synchronization can 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, the nodes cannot become completely synchronized. The "leading" transmitter does not necessarily win the arbitration; therefore, the receivers must 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 must synchronize to the receiver that "takes the lead" in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration are 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 10 bits). These summarized differences cannot be longer than the SJW, limiting the oscillator's tolerance range.

The examples in [Figure 16-16](#) 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 16-16. 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 the edge 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 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 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 the 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 the state machine cannot subsequently redefine that time quantum of Phase\_Seg2 where the edge occurs to be the Sync\_Seg.

The examples in Figure 16-17 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.

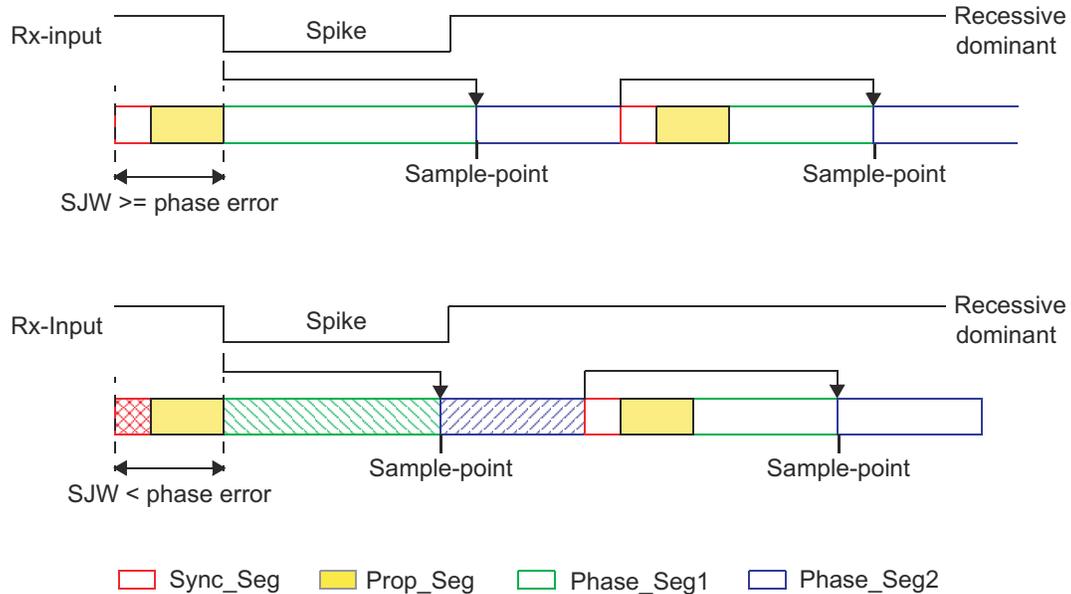


Figure 16-17. Filtering of Short Dominant Spikes

### 16.11.1.4 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:

$$(1 - df) * f_{nom} \leq f_{osc} \leq (1 + df) * f_{nom}$$

depends on the proportions of Phase\_Seg1, Phase\_Seg2, SJW, and the bit time. The maximum tolerance  $df$  is defined by two conditions (both must be met):

$$df \leq \frac{\min(Tseg1, Tseg2)}{2((13 \times bit\ time) - Tseg2)}$$

$$df \leq \frac{SJW}{20 \times bit\_time}$$

You must consider that SJW cannot be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that can 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 for short bit times; the combination can be used for bit rates of up to 125kBit/s (bit time = 8 $\mu$ s) with a bus length of 40 meters.

### 16.11.2 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 16-18](#)).

In this bit timing register, the components TSEG1, TSEG2, SJW, and BRP are programmed to a numerical value that is one less than the 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 either:

- (programmed values) [TSEG1 + TSEG2 + 3]  $t_q$
- (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. Loading and shifting is controlled by the BSP.

The BSP translates messages into frames and conversely. The BSP 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 cannot be longer than  $2 t_q$ . The IPT length is the lower limit of the programmed length of Phase\_Seg2. In case of a synchronization, Phase\_Seg2 can be shortened to a value less than IPT, which does not affect bus timing.

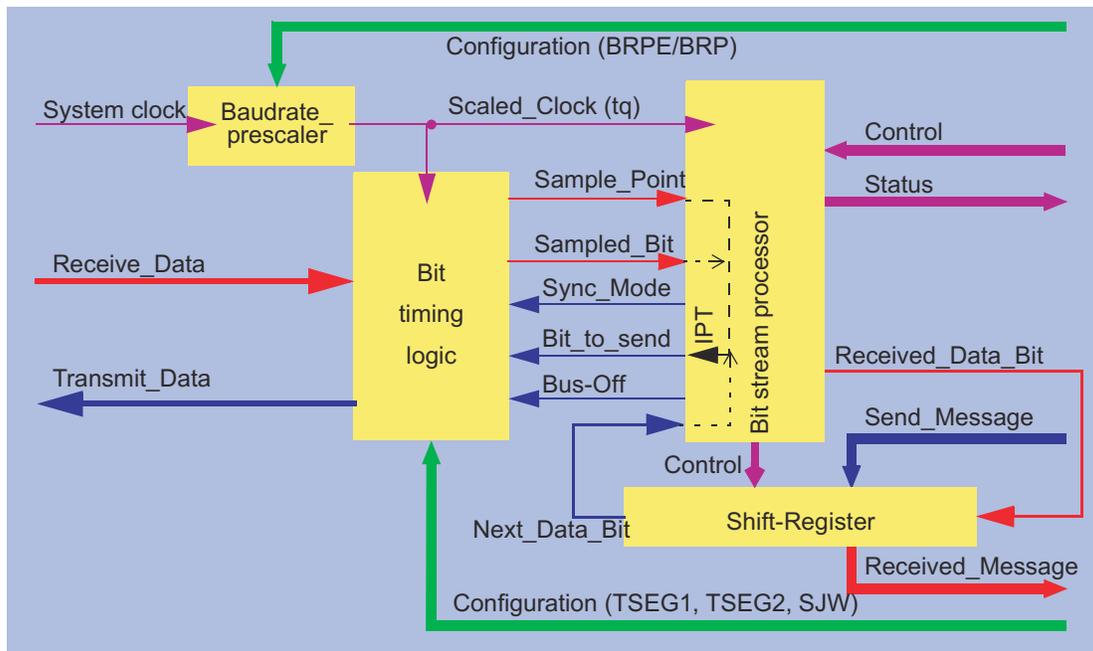


Figure 16-18. Structure of the CAN Core's CAN Protocol Controller

### 16.11.2.1 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/Bit rate) must be an integer multiple of the CAN clock period.

---

#### Note

8MHz is the minimum CAN clock frequency required to operate the CAN at a bit rate of 1MBit/s.

---

The bit time can 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 can 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. The 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,  $\text{Phase\_Seg2} = \text{Phase\_Seg1}$ ; else,  $\text{Phase\_Seg2} = \text{Phase\_Seg1} + 1$ .

The minimum nominal length of Phase\_Seg2 has to be regarded as well. Phase\_Seg2 cannot be shorter than the Information Processing Time of any node in the network, which is device dependent and can be in the range of  $[0 \text{ to } 2] t_q$ .

The length of the synchronization jump width is set to the 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 16.11.1.4](#).

If more than one configurations are possible to reach a certain bit rate, choose the configuration that 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 oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation can show that bus length or bit rate has to be decreased or that the oscillator frequency stability has to be increased to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing register:

$$(\text{Phase\_Seg2}-1) \& (\text{Phase\_Seg1} + \text{Prop\_Seg} - 1) \& (\text{SynchronizationJumpWidth} - 1) \& (\text{Prescaler} - 1)$$

### 16.11.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN\_CLK is 10MHz, BRP is 0, the bit rate is 1MBit/s.

$t_q$	100ns =	$t_{CAN\_CLK}$
delay of bus driver	90ns =	
delay of receiver circuit	40ns =	
delay of bus line (40m)	220ns =	
$t_{Prop}$	700ns =	$2 \cdot \text{delays} = 7 \cdot t_q$
$t_{SJW}$	100ns =	$1 \cdot t_q$
$t_{TSeg1}$	800ns =	$t_{Prop} + t_{SJW}$
$t_{TSeg2}$	100ns =	Information Processing Time + $1 \cdot t_q$
$t_{Sync-Seg}$	100ns =	$1 \cdot t_q$
bit time	1000ns =	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
tolerance for CAN_CLK	0.35% =	$\frac{\min(Tseg1, Tseg2)}{2((13 \times \text{bit time}) - Tseg2)}$
		$= \frac{0.1 \mu s}{2((13 \times 1 \mu s) - 0.1 \mu s)}$

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 = 0x0000 0700.

### 16.11.2.3 Example for Bit Timing at Low Baudrate

In this example, the frequency of CAN\_CLK is 2MHz, BRP is 1, the bit rate is 100KBit/s.

$t_q$	1 $\mu$ s =	$2 \cdot t_{CAN\_CLK}$
delay of bus driver	200ns =	
delay of receiver circuit	80ns =	
delay of bus line (40m)	220ns =	
$t_{Prop}$	1 $\mu$ s =	$1 \cdot t_q$
$t_{SJW}$	4 $\mu$ s =	$4 \cdot t_q$
$t_{TSeg1}$	5 $\mu$ s =	$t_{Prop} + t_{SJW}$
$t_{TSeg2}$	4 $\mu$ s =	Information Processing Time + $4 \cdot t_q$
$t_{Sync-Seg}$	1 $\mu$ s =	$1 \cdot t_q$
bit time	10 $\mu$ s =	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
tolerance for CAN_CLK	1.58% =	$\frac{\min(Tseg1, Tseg2)}{2((13 \times \text{bit time}) - Tseg2)}$
		$= \frac{4 \mu s}{2((13 \times 10 \mu s) - 4 \mu s)}$

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 = 0x0000 34C1.

## 16.12 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the Message RAM. There are two interface register sets for read and write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore the data transfer from the IF1/IF2 registers to the Message RAM requires the message handler to perform a read-modify-write cycle. First those parts of the message object that are not to be changed are read from the Message RAM into the Interface Register set, and after the update the whole content of the Interface Register set is written into the message object.

After the partial write of a message object, those parts of the Interface Register set that are not selected in the Command Register are 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 that are not selected in the Command Register are 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 16.13.1](#)) or parts of the message object can 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, maintains the data consistency of the CAN message.

There is one condition that can cause a write access to the message RAM to be lost. If `MsgVal = 1` for the message object that is accessed and CAN communication is ongoing, a transfer from the IFx register to message RAM can be lost. The reason this can happen is that the IFx register write to the message RAM occurs in between a read-modify-write access of the Host Message Handler when in the process of receiving a message for the same message object.

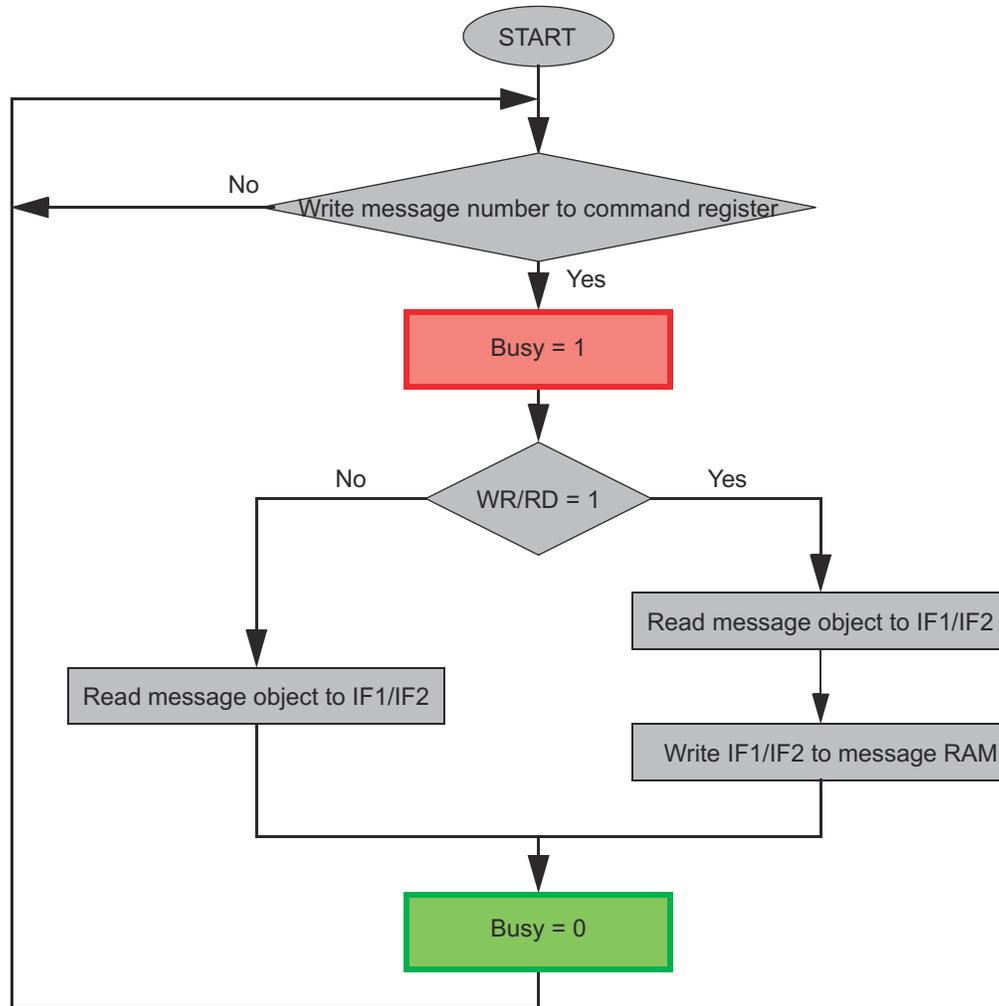
To avoid this issue with receive mail boxes, reset `MsgVal` before changing any of the following: `Id28-0`, `Xtd`, `Dir`, `DLC3-0`, `RxIE`, `TxIE`, `RmtEn`, `EoB`, `Umask`, `Msk28-0`, `MXtd`, and `MDir`.

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`, can be changed without fear of losing a write to the message RAM.

### 16.12.1 Message Interface Register Sets 1 and 2 (IF1 and IF2)

The IF1 and IF2 register sets allow data transfers to and from the message objects. The IFxCMD register for an interface control the direction of the data transfer. If the IFxCMD register is set to write, then the message object fields selected by the IFxCMD register are overwritten by values taken from the other IFx registers. If the IFxCMD register is set to read, then the message object fields selected by the IFxCMD register is copied from the message object to the other IFx registers. The interfaces allow for transfers of a complete message object as well as individual parts. The transfer begins with the desired message object number is written to bits 7:0 of the IFxCMD register.

When the CPU initiates a data transfer between the IF1/IF2 registers and Message RAM, the message handler sets the Busy bit in the respective Command Register to 1. After the transfer has completed, the Busy bit is set back to 0 (see [Figure 16-19](#)).



**Figure 16-19. Data Transfer Between IF1 / IF2 Registers and Message RAM**

### 16.12.2 Message Interface Register Set 3 (IF3)

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The automatic update functionality can be programmed for each message object (see the IF3 Update Enable register).

All valid message objects in Message RAM that are configured for automatic update are checked for active NewDat flags. If such a message object is found, the message objects are 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 is reset by a transfer to IF3.

**Note**

The IF3 register set cannot be used for transferring data into message objects.

## 16.13 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 using 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.

### 16.13.1 Structure of Message Objects

Figure 16-20 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 16-20. Structure of a Message Object**

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxE	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 16-5. 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 can 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.
UMask		Use Acceptance Mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering. Note: If the UMask bit is set to 1, the message object's mask bits are programmed during initialization of the message object before MsgVal is set to 1.
ID[28:0]		Message Identifier
	ID[28:0]	29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits

**Table 16-5. Message Object Field Descriptions (continued)**

Name	Value	Description
Msk[28:0]	0	Identifier Mask The corresponding bit in the message identifier is not used for acceptance filtering (don't care).
	1	The corresponding bit in the message identifier is used for acceptance filtering. Note: The bit functionality in the DCAN module is the opposite of the Local Acceptance Mask bit functionality in the eCAN module found in older C28x devices, where a 1 means the corresponding bit is not used for filtering, and a 0 means the bit is used.
Xtd	0	Extended Identifier The 11-bit ("standard") identifier is used for this message object.
	1	The 29-bit ("extended") identifier is used for this message object.
MXtd	0	Mask Extended Identifier The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering. Note: When 11-bit ("standard") Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir	0	Message Direction Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
MDir	0	Mask Message Direction The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB	0	End of Block The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block. Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
NewDat	0	New Data No new data has been written into the data bytes of this message object by the message handler since the last time when this flag was cleared by the CPU.
	1	The message handler or the CPU has written new data into the data bytes of this message object.
MsgLst	0	Message Lost (only valid for Message Objects with direction = receive) No message was lost since the last time when this bit was reset by the CPU.
	1	The message handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE	0	Receive Interrupt Enable IntPnd is not triggered after the successful reception of a frame.
	1	IntPnd is triggered after the successful reception of a frame.
TxIE	0	Transmit Interrupt Enable IntPnd is not triggered after the successful transmission of a frame.
	1	IntPnd is triggered after the successful transmission of a frame.
IntPnd	0	Interrupt Pending This message object is not the source of an interrupt.
	1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register point to this message object, if there is no other interrupt source with higher priority.

**Table 16-5. Message Object Field Descriptions (continued)**

Name	Value	Description
RmtEn	0	Remote Enable At the reception of a remote frame, TxRqst is not changed.
	1	At the reception of a remote frame, TxRqst is set. Note: See <a href="#">Section 16.10.8</a> for details on the setup of RmtEn and UMask for remote frames.
TxRqst	0	Transmit Request This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]	0-8	Data length code Data frame has 0-8 data bytes.
	9-15	Data frame has 8 data bytes. Note: The data length code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, the message handler writes 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, the message handler writes 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 can be overwritten by undefined values.

### 16.13.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

$$\text{Message RAM base address} + (\text{message object number}) * 0x20$$

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, and so on.

#### Note

A 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 can overwrite an implemented message object.

Message Object number 1 has the highest priority.

**Table 16-6. Message RAM Addressing in Debug Mode**

Message Object Number	Offset From Base Address	Word Number	Debug Mode <sup>(1)</sup>
last implemented (here:32)	0x0000	1	Parity
	0x0004	2	MXtd,MDir,Mask
	0x0008	3	Xtd,Dir,ID
	0x000C	4	Ctrl
	0x0010	5	Data Bytes 3-0
	0x0014	6	Data Bytes 7-4
1	0x0020	1	Parity
	0x0024	2	MXtd,MDir,Mask
	0x0028	3	Xtd,Dir,ID
	0x002C	4	Ctrl
	0x0030	5	Data Bytes 3-0
	0x0034	6	Data Bytes 7-4
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 16.13.3](#).

### 16.13.3 Message RAM Representation in Debug Mode

In debug mode, the Message RAM is memory-mapped. This allows the external debug unit to access the Message RAM.

#### Note

During debug mode, the Message RAM cannot be accessed using the IFx register sets.

**Figure 16-21. Message RAM Representation in Debug Mode**

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

## 16.14 Software

### 16.14.1 CAN Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/can

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 16.14.1.1 CAN External Loopback

FILE: can\_ex1\_loopback.c

This example shows the basic setup of CAN to transmit and receive messages on the CAN bus. The CAN peripheral is configured to transmit messages with a specific CAN ID. A message is then transmitted once per second, using a simple delay loop for timing. The message that is sent is a 2 byte message that contains an incrementing pattern.

This example sets up the CAN controller in External Loopback test mode. Data transmitted is visible on the CANTXA pin and is received internally back to the CAN Core. Please refer to details of the External Loopback Test Mode in the CAN Chapter in the Technical Reference Manual. Refer to [Programming Examples and Debug Strategies for the DCAN Module](#) for useful information about this example

##### External Connections

- None.

##### Watch Variables

- msgCount - A counter for the number of successful messages received
- txMsgData - An array with the data being sent
- rxMsgData - An array with the data that was received

#### 16.14.1.2 CAN External Loopback with Interrupts

FILE: can\_ex2\_loopback\_interrupts.c

This example shows the basic setup of CAN to transmit and receive messages on the CAN bus. The CAN peripheral is configured to transmit messages with a specific CAN ID. A message is then transmitted once per second, using a simple delay loop for timing. The message that is sent is a 4 byte message that contains an incrementing pattern. A CAN interrupt handler is used to confirm message transmission and count the number of messages that have been sent.

This example sets up the CAN controller in External Loopback test mode. Data transmitted is visible on the CANTXA pin and is received internally back to the CAN Core. Please refer to details of the External Loopback Test Mode in the CAN Chapter in the Technical Reference Manual. Refer to [Programming Examples and Debug Strategies for the DCAN Module](#) for useful information about this example

##### External Connections

- None.

##### Watch Variables

- txMsgCount - A counter for the number of messages sent
- rxMsgCount - A counter for the number of messages received
- txMsgData - An array with the data being sent
- rxMsgData - An array with the data that was received
- errorFlag - A flag that indicates an error has occurred

#### 16.14.1.3 CAN Transmit and Receive Configurations

FILE: can\_ex5\_transmit\_receive.c

This example shows the basic setup of CAN to transmit or receive messages on the CAN bus with a specific Message ID. The CAN Controller is configured according to the selection of the define.

When the TRANSMIT define is selected, the CAN Controller acts as a Transmitter and sends data to the second CAN Controller connected externally. If TRANSMIT is not defined the CAN Controller acts as a Receiver and waits for message to be transmitted by the External CAN Controller. Refer to [Programming Examples and Debug Strategies for the DCAN Module](#) for useful information about this example

CAN modules on the device need to be connected to via CAN transceivers. *Hardware Required*

- A C2000 board with CAN transceiver.

#### External Connections

- ControlCARD CANA is on DEVICE\_GPIO\_PIN\_CANTXA (CANTXA)
- and DEVICE\_GPIO\_PIN\_CANRXA (CANRXA)

#### Watch Variables Transmit \Configuration

- MSGCOUNT - Adjust to set the number of messages
- txMsgCount - A counter for the number of messages sent
- txMsgData - An array with the data being sent
- errorFlag - A flag that indicates an error has occurred
- rxMsgCount - Has the initial value as No. of Messages to be received and decrements with each message.

#### 16.14.1.4 CAN Error Generation Example

FILE: can\_ex6\_error\_generation.c

This example demonstrates the ways of handling CAN Error conditions. It generates the CAN Packets and sends them over GPIO. It is looped back externally to be received in CAN module. The CAN Interrupt service routine reads the Error status and demonstrates how different Error conditions can be detected.

Change ERR\_CFG define to the different Error Scenarios and run the example. The corresponding Error Flag will be set in status variable of canISR() routine. Uses a CPU Timer (Timer 0) for periodic timer interrupt of CANBITRATE uSec. On the Timer interrupt it sends the required CAN Frame type with the specified error conditions. CAN modules on the device need to be connected to via CAN transceivers. Please refer to the application note titled "Configurable Error Generator for Controller Area Network" at [Configurable Error Generator for Controller Area Network](#) for further details on this example.

#### External Connections

- ControlCARD GPIOTX\_PIN should be connected to
- DEVICE\_GPIO\_PIN\_CANRXA (CANRXA)

#### Watch Variables Transmit \Configuration

- status - variable in canISR for checking error Status

#### 16.14.1.5 CAN Remote Request Loopback

FILE: can\_ex7\_loopback\_tx\_rx\_remote\_frame.c

This example shows the basic setup of CAN in order to transmit a remote frame and get a response for the remote frame and store it in a receive Object. The CAN peripheral is configured to transmit remote request frame and a remote answer frame messages with a specific CAN ID. Message object 3 is configured to transmit a remote request. Message object 2 is configured as a remote answer object with filter mask such that it accepts remote frame with any message ID and transmit's remote answer with message ID 7 and data length 8. Message object 1 is configured as a received object with filter message ID 7 so as to store the remote answer data transmitted by message object 2.

This example sets up the CAN controller in External Loopback test mode. Data transmitted is visible on the CANTXA pin and is received internally back to the CAN Core. Please refer to details of the External Loopback Test Mode in the CAN Chapter in the Technical Reference Manual.

#### External Connections

- None.

### Watch Variables

- txMsgData - An array with the data being sent
- rxMsgData - An array with the data that was received

#### 16.14.1.6 CAN example that illustrates the usage of Mask registers

FILE: can\_ex8\_mask.c

This example initializes CAN module A for Reception. When a frame with a matching filter criterion is received, the data is copied in mailbox 1 and LED is toggled a few times and the code gets ready for the next frame. If a message of any other MSGID is received, an ACK is provided. Completion of reception is determined by polling CAN\_NDAT\_21 register. No interrupts are used. Refer to [Programming Examples and Debug Strategies for the DCAN Module](#) for useful information about this example

### Hardware Required

- An external CAN node that transmits to CAN-A on the C2000 MCU

### Watch Variables

- rxMsgCount - A counter for the number of messages received
- rxMsgData - An array with the data that was received

## 16.15 CAN Registers

This section describes the Controller Area Network registers.

### 16.15.1 CAN Base Address Table

**Table 16-7. CAN Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
CanaRegs	CAN_REGS	CANA_BASE	0x0004_8000	YES

## 16.15.2 CAN\_REGS Registers

Table 16-8 lists the memory-mapped registers for the CAN\_REGS registers. All register offset addresses not listed in Table 16-8 should be considered as reserved locations and the register contents should not be modified.

**Table 16-8. CAN\_REGS Registers**

Offset (x8)	Offset (x16)	Acronym	Register Name	Write Protection	Section
0h	0h	CAN_CTL	CAN Control Register		<a href="#">Go</a>
8h	4h	CAN_ES	Error and Status Register		<a href="#">Go</a>
10h	8h	CAN_ERRC	Error Counter Register		<a href="#">Go</a>
18h	Ch	CAN_BTR	Bit Timing Register		<a href="#">Go</a>
20h	10h	CAN_INT	Interrupt Register		<a href="#">Go</a>
28h	14h	CAN_TEST	Test Register		<a href="#">Go</a>
38h	1Ch	CAN_PERR	CAN Parity Error Code Register		<a href="#">Go</a>
80h	40h	CAN_RAM_INIT	CAN RAM Initialization Register		<a href="#">Go</a>
A0h	50h	CAN_GLB_INT_EN	CAN Global Interrupt Enable Register		<a href="#">Go</a>
A8h	54h	CAN_GLB_INT_FLG	CAN Global Interrupt Flag Register		<a href="#">Go</a>
B0h	58h	CAN_GLB_INT_CLR	CAN Global Interrupt Clear Register		<a href="#">Go</a>
100h	80h	CAN_ABOTR	Auto-Bus-On Time Register		<a href="#">Go</a>
108h	84h	CAN_TXRQ_X	CAN Transmission Request Register		<a href="#">Go</a>
110h	88h	CAN_TXRQ_21	CAN Transmission Request 2_1 Register		<a href="#">Go</a>
130h	98h	CAN_NDAT_X	CAN New Data Register		<a href="#">Go</a>
138h	9Ch	CAN_NDAT_21	CAN New Data 2_1 Register		<a href="#">Go</a>
158h	ACh	CAN_IPEN_X	CAN Interrupt Pending Register		<a href="#">Go</a>
160h	B0h	CAN_IPEN_21	CAN Interrupt Pending 2_1 Register		<a href="#">Go</a>
180h	C0h	CAN_MVAL_X	CAN Message Valid Register		<a href="#">Go</a>
188h	C4h	CAN_MVAL_21	CAN Message Valid 2_1 Register		<a href="#">Go</a>
1B0h	D8h	CAN_IP_MUX21	CAN Interrupt Multiplexer 2_1 Register		<a href="#">Go</a>
200h	100h	CAN_IF1CMD	IF1 Command Register		<a href="#">Go</a>
208h	104h	CAN_IF1MSK	IF1 Mask Register		<a href="#">Go</a>
210h	108h	CAN_IF1ARB	IF1 Arbitration Register		<a href="#">Go</a>
218h	10Ch	CAN_IF1MCTL	IF1 Message Control Register		<a href="#">Go</a>
220h	110h	CAN_IF1DATA	IF1 Data A Register		<a href="#">Go</a>
228h	114h	CAN_IF1DATB	IF1 Data B Register		<a href="#">Go</a>
240h	120h	CAN_IF2CMD	IF2 Command Register		<a href="#">Go</a>
248h	124h	CAN_IF2MSK	IF2 Mask Register		<a href="#">Go</a>
250h	128h	CAN_IF2ARB	IF2 Arbitration Register		<a href="#">Go</a>
258h	12Ch	CAN_IF2MCTL	IF2 Message Control Register		<a href="#">Go</a>
260h	130h	CAN_IF2DATA	IF2 Data A Register		<a href="#">Go</a>
268h	134h	CAN_IF2DATB	IF2 Data B Register		<a href="#">Go</a>
288h	144h	CAN_IF3MSK	IF3 Mask Register		<a href="#">Go</a>
290h	148h	CAN_IF3ARB	IF3 Arbitration Register		<a href="#">Go</a>
298h	14Ch	CAN_IF3MCTL	IF3 Message Control Register		<a href="#">Go</a>
2A0h	150h	CAN_IF3DATA	IF3 Data A Register		<a href="#">Go</a>
2A8h	154h	CAN_IF3DATB	IF3 Data B Register		<a href="#">Go</a>
2C0h	160h	CAN_IF3UPD	IF3 Update Enable Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. [Table 16-9](#) shows the codes that are used for access types in this section.

**Table 16-9. CAN\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 16.15.2.1 CAN\_CTL Register (Offset (x8) = 0h, Offset (x16) = 0h) [Reset = 00001401h]

CAN\_CTL is shown in [Figure 16-22](#) and described in [Table 16-10](#).

Return to the [Summary Table](#).

This register is used for configuring the CAN module in terms of interrupts, parity, debug-mode behavior etc.

**Figure 16-22. CAN\_CTL Register**

31	30	29	28	27	26	25	24
RESERVED						RESERVED	RESERVED
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/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h
15	14	13	12	11	10	9	8
SWR	RESERVED	PMD				ABO	IDS
R-0/W1C-0h	R-0h	R/W-5h				R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
Test	CCE	DAR	RESERVED	EIE	SIE	IE0	Init
R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h

**Table 16-10. CAN\_CTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	Reserved
25	RESERVED	R/W	0h	Reserved
24	RESERVED	R/W	0h	Reserved
23-21	RESERVED	R	0h	Reserved
20	RESERVED	R/W	0h	Reserved
19	RESERVED	R/W	0h	Reserved
18	RESERVED	R/W	0h	Reserved
17	IE1	R/W	0h	Interrupt line 1 Enable 0 CANINT1 is disabled. 1 CANINT1 is enabled. Interrupts will assert CANINT1 line to 1 line remains active until pending interrupts are processed. Reset type: SYSRSn
16	INITDBG	R	0h	Debug Mode Status Bit: This bit indicates the internal init state for a debug access 0 Not in debug mode, or debug mode requested but not entered. 1 Debug mode requested and internally entered the CAN module is ready for debug accesses. Reset type: SYSRSn

**Table 16-10. CAN\_CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	SWR	R-0/W1C	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. Note: This bit is write-protected by Init bit. If module is reset using the SWR bit, no user configuration is lost. Only status bits get reset along with logic which needs to be reset for the next CAN transaction. If module is reset using SOFTPRES register, entire module will get reset, including configuration registers. Reset type: SYSRSn
14	RESERVED	R	0h	Reserved
13-10	PMD	R/W	5h	Parity on/off 0101 Parity function disabled Any other value - Parity function enabled Reset type: SYSRSn
9	ABO	R/W	0h	Auto-Bus-On Enable 0 The Auto-Bus-On feature is disabled 1 The Auto-Bus-On feature is enabled Reset type: SYSRSn
8	IDS	R/W	0h	Interruption Debug Support Enable 0 When Debug mode is requested, the CAN module will wait for a started transmission or reception to be completed before entering Debug mode 1 When Debug mode is requested, the CAN module will interrupt any transmission or reception, and enter Debug mode immediately. Reset type: SYSRSn
7	Test	R/W	0h	Test Mode Enable 0 Disable Test Mode (Normal operation) 1 Enable Test Mode Reset type: SYSRSn
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). Reset type: SYSRSn
5	DAR	R/W	0h	Disable Automatic Retransmission 0 Automatic Retransmission of 'not successful' messages enabled. 1 Automatic Retransmission disabled. Reset type: SYSRSn
4	RESERVED	R	0h	Reserved
3	EIE	R/W	0h	Error Interrupt Enable 0 Disabled - PER, BOff and EWarn bits cannot generate an interrupt. 1 Enabled - PER, BOff and EWarn bits can generate an interrupt at CANINT0 line and affect the Interrupt Register. Reset type: SYSRSn
2	SIE	R/W	0h	Status Change Interrupt Enable 0 Disabled - RxOk, TxOk and LEC bits cannot generate an interrupt. 1 Enabled - RxOk, TxOk and LEC can generate an interrupt on the CANINT0 line Reset type: SYSRSn

**Table 16-10. CAN\_CTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	IE0	R/W	0h	Interrupt line 0 Enable 0 CANINT0 is disabled. 1 CANINT0 is enabled. Interrupts will assert CANINT0 line to 1 line remains active until pending interrupts are processed. Reset type: SYSRSn
0	Init	R/W	1h	Initialization Mode This bit is used to keep the CAN module inactive during bit timing configuration and message RAM initialization. It is set automatically during a bus off event. Clearing this bit will not shorten the bus recovery time. 0 CAN module processes messages normally 1 CAN module ignores bus activity Reset type: SYSRSn

### 16.15.2.2 CAN\_ES Register (Offset (x8) = 8h, Offset (x16) = 4h) [Reset = 0000007h]

CAN\_ES is shown in [Figure 16-23](#) and described in [Table 16-11](#).

Return to the [Summary Table](#).

This register indicates error conditions, if any, of the CAN module. Interrupts are generated by PER, BOff and EWarn bits (if EIE bit in CAN Control Register is set) and by RxOk, TxOk, and LEC bits (if SIE bit in CAN Control Register is set). A change of bit EPass will not generate an Interrupt.

Reading the Error and Status Register clears the PER, RxOk and TxOk bits and sets the LEC to value '7'. Additionally, the Status Interrupt value (0x8000) in the Interrupt Register will be replaced by the next lower priority interrupt value.

For debug support, the auto clear functionality of Error and Status Register (clear of status flags by read) is disabled when in Debug/Suspend mode.

**Figure 16-23. CAN\_ES Register**

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

**Table 16-11. CAN\_ES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10	RESERVED	R	0h	Reserved
9	RESERVED	R	0h	Reserved
8	PER	R	0h	Parity Error Detected: This bit will be reset after the CPU reads the register. 0 No parity error has been detected since last read access. 1 The parity check mechanism has detected a parity error in the Message RAM. Reset type: SYSRSn
7	BOff	R	0h	Bus-off Status Bit: 0 The CAN module is not in Bus-Off state. 1 The CAN module is in Bus-Off state. Reset type: SYSRSn
6	EWarn	R	0h	Warning State Bit: 0 Both error counters are below the error warning limit of 96. 1 At least one of the error counters has reached the error warning limit of 96. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 16-11. 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. This bit will be set independent of the result of acceptance filtering.</p> <p>Reset type: SYSRSn</p>
3	TxOk	R	0h	<p>Transmission status Bit: This bit indicates the status of transmission. The bit will be reset after the CPU reads the register.</p> <p>0 No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by CAN internal events.</p> <p>1 A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was cleared by a read access of the CPU.</p> <p>Reset type: SYSRSn</p>
2-0	LEC	R	7h	<p>Last Error Code</p> <p>The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. This field will be reset to '7' whenever the CPU reads the register.</p> <p>0 No Error</p> <p>1 Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.</p> <p>2 Form Error: A fixed format part of a received frame has the wrong format.</p> <p>3 Ack Error: The message this CAN Core transmitted was not acknowledged by another node.</p> <p>4 Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.</p> <p>5 Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).</p> <p>6 CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).</p> <p>7 No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-initializes the LEC to value '7'.</p> <p>Reset type: SYSRSn</p>

### 16.15.2.3 CAN\_ERRC Register (Offset (x8) = 10h, Offset (x16) = 8h) [Reset = 0000000h]

CAN\_ERRC is shown in [Figure 16-24](#) and described in [Table 16-12](#).

Return to the [Summary Table](#).

This register reflects the value of the Transmit and Receive error counters

**Figure 16-24. CAN\_ERRC Register**

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

**Table 16-12. CAN\_ERRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	RP	R	0h	Receive Error Passive 0 The Receive Error Counter is below the error passive level. 1 The Receive Error Counter has reached the error passive level as defined in the CAN Specification. Reset type: SYSRSn
14-8	REC	R	0h	Receive Error Counter Actual state of the Receive Error Counter (values from 0 to 127). Reset type: SYSRSn
7-0	TEC	R	0h	Transmit Error Counter Actual state of the Transmit Error Counter. (values from 0 to 255). Reset type: SYSRSn

### 16.15.2.4 CAN\_BTR Register (Offset (x8) = 18h, Offset (x16) = Ch) [Reset = 00002301h]

CAN\_BTR is shown in [Figure 16-25](#) and described in [Table 16-13](#).

Return to the [Summary Table](#).

This register is used to configure the bit-timing parameters for the CAN module. This register is only writable if CCE and Init bits in the CAN Control Register are set.

The CAN bit time may be programmed in the range of 8 to 25 time quanta.

The CAN time quantum may be programmed in the range of 1 to 1024 CAN\_CLK periods.

**Figure 16-25. CAN\_BTR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED				BRPE			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED	TSEG2			TSEG1			
R-0h		R/W-2h		R/W-3h			
7	6	5	4	3	2	1	0
SJW		BRP					
R/W-0h		R/W-1h					

**Table 16-13. CAN\_BTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-16	BRPE	R/W	0h	Baud Rate Prescaler Extension Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
15	RESERVED	R	0h	Reserved
14-12	TSEG2	R/W	2h	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
11-8	TSEG1	R/W	3h	Time segment before the sample point Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn
7-6	SJW	R/W	0h	Synchronization Jump Width Valid programmed values are 0 to 3. The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn

**Table 16-13. CAN\_BTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5-0	BRP	R/W	1h	Baud Rate Prescaler- Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63. The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1. Note: This bit is Write Protected by CCE bit. Reset type: SYSRSn

### 16.15.2.5 CAN\_INT Register (Offset (x8) = 20h, Offset (x16) = 10h) [Reset = 0000000h]

CAN\_INT is shown in [Figure 16-26](#) and described in [Table 16-14](#).

Return to the [Summary Table](#).

This register is used to identify the source of the interrupt(s).

**Figure 16-26. CAN\_INT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								INT1ID								INT0ID															
R-0h								R-0h								R-0h															

**Table 16-14. CAN\_INT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	INT1ID	R	0h	Interrupt 1 Cause 0x00 No interrupt is pending. 0x01-0x20 Number of message object (mailbox) which caused the interrupt. 0x21-0xFF Unused. If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. Note: The CANINT1 interrupt line remains active until INT1ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. A message interrupt is cleared by clearing the mailbox's IntPnd bit. Among the message interrupts, the mailbox's interrupt priority decreases with increasing message number. Reset type: SYSRSn
15-0	INT0ID	R	0h	Interrupt 0 Cause 0x0000 - No interrupt is pending. 0x0001 - 0x0020 - Number of message object which caused the interrupt. 0x0021 - 0x7FFF - Unused. 0x8000 - Error and Status Register value is not 0x07. 0x8001 - 0xFFFF - Unused. If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. Note: The CANINT0 interrupt line remains active until INT0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. Reset type: SYSRSn

### 16.15.2.6 CAN\_TEST Register (Offset (x8) = 28h, Offset (x16) = 14h) [Reset = 0000000h]

CAN\_TEST is shown in [Figure 16-27](#) and described in [Table 16-15](#).

Return to the [Summary Table](#).

This register is used to configure the various test options supported. For all test modes, the Test bit in CAN Control Register needs to be set to one. If Test bit is set, the RDA, EXL, Tx1, Tx0, LBack and Silent bits are writable. Bit Rx monitors the state of CANRX pin and therefore is only readable. All Test Register functions are disabled when Test bit is cleared.

Note: Setting Tx[1:0] other than '00' will disturb message transfer.

Note: When the internal loop back mode is active (bit LBack is set), bit EXL will be ignored.

**Figure 16-27. CAN\_TEST Register**

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

**Table 16-15. CAN\_TEST Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	RDA	R/W	0h	RAM Direct Access Enable: 0 Normal Operation. 1 Direct access to the RAM is enabled while in Test Mode. Reset type: SYSRSn
8	EXL	R/W	0h	External Loop Back Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn
7	RX	R	0h	Monitors the actual value of the CANRX pin: 0 The CAN bus is dominant. 1 The CAN bus is recessive. Reset type: SYSRSn
6-5	TX	R/W	0h	Control of CANTX pin: 00 Normal operation, CANTX is controlled by the CAN Core. 01 Sample Point can be monitored at CANTX pin. 10 CANTX pin drives a dominant value. 11 CANTX pin drives a recessive value. Reset type: SYSRSn
4	LBACK	R/W	0h	Loop Back Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn

**Table 16-15. CAN\_TEST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	SILENT	R/W	0h	Silent Mode: 0 Disabled. 1 Enabled. Reset type: SYSRSn
2-0	RESERVED	R	0h	Reserved

### 16.15.2.7 CAN\_PERR Register (Offset (x8) = 38h, Offset (x16) = 1Ch) [Reset = 0000XXXh]

CAN\_PERR is shown in [Figure 16-28](#) and described in [Table 16-16](#).

Return to the [Summary Table](#).

This register indicates the Word/Mailbox number where a parity error has been detected. If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism

it must be reset by reading the Error and Status Register. In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected. If more than one word with a parity error was detected, the highest word number with a parity error will be displayed. After a parity error has been detected, the register will hold the last error code until power is removed.

**Figure 16-28. 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-X					R-X					

**Table 16-16. CAN\_PERR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Reserved
10-8	WORD_NUM	R	X	0x01-0x05 Word number where parity error has been detected. RDA word number (1 to 5) of the mailbox (according to the Message RAM representation in RDA mode). Reset type: SYSRSn
7-0	MSG_NUM	R	X	0x01-0x21 Mailbox number where parity error has been detected Reset type: SYSRSn

### 16.15.2.8 CAN\_RAM\_INIT Register (Offset (x8) = 80h, Offset (x16) = 40h) [Reset = 0000005h]

CAN\_RAM\_INIT is shown in [Figure 16-29](#) and described in [Table 16-17](#).

Return to the [Summary Table](#).

This register is used to initialize the Mailbox RAM. It clears the entire mailbox RAM, including the MsgVal bits.

**Figure 16-29. CAN\_RAM\_INIT Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		RAM_INIT_DONE	CAN_RAM_INIT	KEY3	KEY2	KEY1	KEY0
R-0h		R-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-1h

**Table 16-17. CAN\_RAM\_INIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	0h	Reserved
5	RAM_INIT_DONE	R	0h	CAN Mailbox RAM initialization status: 0 Read: Initialization is on-going or initialization not initiated. 1 Read: Initialization complete Reset type: SYSRSn
4	CAN_RAM_INIT	R/W	0h	Initiate CAN Mailbox RAM initialization: 0 Read: Initialization complete or initialization not initiated. Write: No action 1 Read: Initialization is on-going Write: Initiate CAN Mailbox RAM initialization. After initialization, this bit will be automatically cleared to 0. Reset type: SYSRSn
3	KEY3	R/W	0h	See Key 0 Reset type: SYSRSn
2	KEY2	R/W	1h	See Key 0 Reset type: SYSRSn
1	KEY1	R/W	0h	See Key 0 Reset type: SYSRSn
0	KEY0	R/W	1h	KEY3-KEY0 should be 1010 for any write to this register to be valid. These bits will be restored to their reset state after the CAN RAM initialization is complete. Reset type: SYSRSn

### 16.15.2.9 CAN\_GLB\_INT\_EN Register (Offset (x8) = A0h, Offset (x16) = 50h) [Reset = 0000000h]

CAN\_GLB\_INT\_EN is shown in [Figure 16-30](#) and described in [Table 16-18](#).

Return to the [Summary Table](#).

This register is used to enable the interrupt lines to the PIE.

**Figure 16-30. CAN\_GLB\_INT\_EN Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						GLBINT1_EN	GLBINT0_EN
R-0h						R/W-0h	R/W-0h

**Table 16-18. CAN\_GLB\_INT\_EN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	GLBINT1_EN	R/W	0h	Global Interrupt Enable for CANINT1 0 CANINT1 does not generate interrupt to PIE 1 CANINT1 generates interrupt to PIE if interrupt condition occurs Reset type: SYSRSn
0	GLBINT0_EN	R/W	0h	Global Interrupt Enable for CANINT0 0 CANINT0 does not generate interrupt to PIE 1 CANINT0 generates interrupt to PIE if interrupt condition occurs Reset type: SYSRSn

**16.15.2.10 CAN\_GLB\_INT\_FLG Register (Offset (x8) = A8h, Offset (x16) = 54h) [Reset = 0000000h]**

 CAN\_GLB\_INT\_FLG is shown in [Figure 16-31](#) and described in [Table 16-19](#).

 Return to the [Summary Table](#).

This register indicates if and when the interrupt line to the PIE is active.

**Figure 16-31. CAN\_GLB\_INT\_FLG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						INT1_FLG	INT0_FLG
R-0h						R-0h	R-0h

**Table 16-19. CAN\_GLB\_INT\_FLG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG	R	0h	CANINT1 Flag 0 No interrupt generated 1 Interrupt is generated due to CANINT1 (refer to CAN Interrupt Status Register for the condition) Reset type: SYSRSn
0	INT0_FLG	R	0h	CANINT0 Flag 0 No interrupt generated 1 Interrupt is generated due to CANINT0 (refer to CAN Interrupt Status Register for the condition) Reset type: SYSRSn

### 16.15.2.11 CAN\_GLB\_INT\_CLR Register (Offset (x8) = B0h, Offset (x16) = 58h) [Reset = 0000000h]

CAN\_GLB\_INT\_CLR is shown in [Figure 16-32](#) and described in [Table 16-20](#).

Return to the [Summary Table](#).

This register is used to clear the interrupt to the PIE.

**Figure 16-32. CAN\_GLB\_INT\_CLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						INT1_FLG_CLR	INT0_FLG_CLR
R-0h						W-0h	W-0h

**Table 16-20. CAN\_GLB\_INT\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	INT1_FLG_CLR	W	0h	Global Interrupt flag clear for CANINT1 0 No effect 1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT1. Reset type: SYSRSn
0	INT0_FLG_CLR	W	0h	Global Interrupt flag clear for CANINT0 0 No effect 1 Write 1 to clear the corresponding bit of the Global Interrupt Flag Register and allow the PIE to receive another interrupt from CANINT0. Reset type: SYSRSn

### 16.15.2.12 CAN\_ABOTR Register (Offset (x8) = 100h, Offset (x16) = 80h) [Reset = 00000000h]

CAN\_ABOTR is shown in [Figure 16-33](#) and described in [Table 16-21](#).

Return to the [Summary Table](#).

This register is used to introduce a variable delay before the Bus-off recovery sequence is started.

**Figure 16-33. CAN\_ABOTR Register**

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

**Table 16-21. CAN\_ABOTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ABO_Time	R/W	0h	Auto-Bus-On Timer Number of clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. 'Clock' refers to the input clock to the CAN module. This function has to be enabled by setting bit ABO in CAN Control Register. 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. NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted. NOTE: During Debug mode, running Auto-Bus-On timer will be paused. Reset type: SYSRSn

### 16.15.2.13 CAN\_TXRQ\_X Register (Offset (x8) = 108h, Offset (x16) = 84h) [Reset = 0000000h]

CAN\_TXRQ\_X is shown in [Figure 16-34](#) and described in [Table 16-22](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Transmission Request 21 Register (CAN\_TXRQ\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the TxRqst bits of these message objects is set, the corresponding bit in this register will be set.

**Figure 16-34. CAN\_TXRQ\_X Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				TxRqstReg2		TxRqstReg1	
R-0h				R-0h		R-0h	

**Table 16-22. CAN\_TXRQ\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	TxRqstReg2	R	0h	Transmit Request Register 2 flag: Bit 2 represents byte 2 of CAN_TXRQ_21. If one or more bits in that byte are set, then bit 2 will be set. Bit 3 represents byte 3 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 3 will be set. Reset type: SYSRSn
1-0	TxRqstReg1	R	0h	Transmit Request Register 1 flag: Bit 0 represents byte 0 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 0 will be set. Bit 1 represents byte 1 of CAN_TXRQ_21 Register. If one or more bits in that byte are set, then bit 1 will be set. Reset type: SYSRSn

### 16.15.2.14 CAN\_TXRQ\_21 Register (Offset (x8) = 110h, Offset (x16) = 88h) [Reset = 00000000h]

CAN\_TXRQ\_21 is shown in [Figure 16-35](#) and described in [Table 16-23](#).

Return to the [Summary Table](#).

This register holds the TxRqst bits of the mailboxes. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific mailbox can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

**Figure 16-35. CAN\_TXRQ\_21 Register**

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

**Table 16-23. CAN\_TXRQ\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TxRqst	R	0h	Transmission Request Bits (for all message objects) 0 No transmission has been requested for this message object. 1 The transmission of this message object is requested and is not yet done. Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,..., Bit 31 is for mailbox 32 Reset type: SYSRSn

### 16.15.2.15 CAN\_NDAT\_X Register (Offset (x8) = 130h, Offset (x16) = 98h) [Reset = 0000000h]

CAN\_NDAT\_X is shown in [Figure 16-36](#) and described in [Table 16-24](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN New Data 21 Register (CAN\_NDAT\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the NewDat bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 16-36. CAN\_NDAT\_X Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				NewDatReg2		NewDatReg1	
R-0h				R-0h		R-0h	

**Table 16-24. CAN\_NDAT\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	NewDatReg2	R	0h	New Data Register 2 flag: Bit 2 represents byte 2 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 2 will be set. Bit 3 represents byte 3 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 3 will be set. Reset type: SYSRSn
1-0	NewDatReg1	R	0h	New Data Register 1 flag: Bit 0 represents byte 0 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 0 will be set. Bit 1 represents byte 1 of CAN_NDAT_21 Register. If one or more bits in that byte are set, then bit 1 will be set. Reset type: SYSRSn

### 16.15.2.16 CAN\_NDAT\_21 Register (Offset (x8) = 138h, Offset (x16) = 9Ch) [Reset = 0000000h]

CAN\_NDAT\_21 is shown in [Figure 16-37](#) and described in [Table 16-25](#).

Return to the [Summary Table](#).

This register holds the NewDat bits of all mailboxes. By reading out the NewDat bits, the CPU can check for which mailboxes the data portion was updated. The NewDat bit of a specific mailbox can be set/reset by the CPU via the IFx 'Message Interface' Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

**Figure 16-37. CAN\_NDAT\_21 Register**

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

**Table 16-25. CAN\_NDAT\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	NewDat	R	0h	New Data Bits (for all message objects) 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: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,..., Bit 31 is for mailbox 32 Reset type: SYSRSn

### 16.15.2.17 CAN\_IPEN\_X Register (Offset (x8) = 158h, Offset (x16) = ACh) [Reset = 0000000h]

CAN\_IPEN\_X is shown in [Figure 16-38](#) and described in [Table 16-26](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Interrupt Pending 21 Register (CAN\_IPEN\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the IntPnd bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 16-38. CAN\_IPEN\_X Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				IntPndReg2		IntPndReg1	
R-0h				R-0h		R-0h	

**Table 16-26. CAN\_IPEN\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	IntPndReg2	R	0h	Interrupt Pending Register 2 flag: Bit 2 represents byte 2 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 2 will be set. Bit 3 represents byte 3 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 3 will be set. Reset type: SYSRSn
1-0	IntPndReg1	R	0h	Interrupt Pending Register 1 flag: Bit 0 represents byte 0 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 0 will be set. Bit 1 represents byte 1 of CAN_IPEN_21 Register. If one or more bits in that byte are set, then bit 1 will be set. Reset type: SYSRSn

### 16.15.2.18 CAN\_IPEN\_21 Register (Offset (x8) = 160h, Offset (x16) = B0h) [Reset = 00000000h]

CAN\_IPEN\_21 is shown in [Figure 16-39](#) and described in [Table 16-27](#).

Return to the [Summary Table](#).

This register holds the IntPnd bits of the mailboxes. By reading out these bits, the CPU can check for pending interrupts in the mailboxes. The IntPnd bit of a specific mailbox can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

**Figure 16-39. CAN\_IPEN\_21 Register**

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

**Table 16-27. CAN\_IPEN\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IntPnd	R	0h	Interrupt Pending bits: This register contains the bits that indicate the pending interrupts in each one of the 32 mailboxes. 0 This mailbox is not the source of an interrupt. 1 This mailbox is the source of an interrupt. Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,..., Bit 31 is for mailbox 32 Reset type: SYSRSn

### 16.15.2.19 CAN\_MVAL\_X Register (Offset (x8) = 180h, Offset (x16) = C0h) [Reset = 0000000h]

CAN\_MVAL\_X is shown in [Figure 16-40](#) and described in [Table 16-28](#).

Return to the [Summary Table](#).

With these bits, the CPU can detect if one or more bits in the CAN Message Valid 2\_1 Register (CAN\_MVAL\_21) is set. Each bit in this register represents a group of eight mailboxes. If at least one of the MsgVal bits of these mailboxes are set, the corresponding bit in this register will be set.

**Figure 16-40. CAN\_MVAL\_X Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				MsgValReg2		MsgValReg1	
R-0h				R-0h		R-0h	

**Table 16-28. CAN\_MVAL\_X Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3-2	MsgValReg2	R	0h	Message Valid Register 2 flag: Bit 2 represents byte 2 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 2 will be set. Bit 3 represents byte 3 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 3 will be set. Reset type: SYSRSn
1-0	MsgValReg1	R	0h	Message Valid Register 1 flag: Bit 0 represents byte 0 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 0 will be set. Bit 1 represents byte 1 of CAN_MVAL_21 Register. If one or more bits in that byte are set, then bit 1 will be set. Reset type: SYSRSn

### 16.15.2.20 CAN\_MVAL\_21 Register (Offset (x8) = 188h, Offset (x16) = C4h) [Reset = 0000000h]

CAN\_MVAL\_21 is shown in [Figure 16-41](#) and described in [Table 16-29](#).

Return to the [Summary Table](#).

This registers hold the MsgVal bits of all mailboxes. By reading out the MsgVal bits, the CPU can check which mailbox is valid. The MsgVal bit of a specific mailbox can be set/reset by the CPU via the IF1/2 'Message Interface' Registers.

**Figure 16-41. CAN\_MVAL\_21 Register**

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

**Table 16-29. CAN\_MVAL\_21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	MsgValReg	R	0h	Message Valid Bits (for all message objects) 0 This message object is ignored by the message handler. 1 This message object is configured and will be considered by the message handler. Note: Bit 0 is for mailbox 1, Bit 1 is for mailbox 2, Bit 2 is for mailbox 3,..., Bit 31 is for mailbox 32 Reset type: SYSRSn

### 16.15.2.21 CAN\_IP\_MUX21 Register (Offset (x8) = 1B0h, Offset (x16) = D8h) [Reset = 0000000h]

CAN\_IP\_MUX21 is shown in [Figure 16-42](#) and described in [Table 16-30](#).

Return to the [Summary Table](#).

The IntMux bit determines for each mailbox, which of the two interrupt lines (CANINT0 or CANINT1) will be asserted when the IntPnd bit of that mailbox is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN Control Register. This will also affect the INT0ID or INT1ID flags in the Interrupt Register.

**Figure 16-42. CAN\_IP\_MUX21 Register**

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

**Table 16-30. CAN\_IP\_MUX21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IntMux	R/W	0h	Interrupt Mux bits: 0 CANINT0 line is active if corresponding IntPnd flag is one. 1 CANINT1 line is active if corresponding IntPnd flag is one. Note: Bit 0 is for mailbox 32, Bit 1 is for mailbox 1, Bit 2 is for mailbox 2,..., Bit 31 is for mailbox 31 Reset type: SYSRSn

### 16.15.2.22 CAN\_IF1CMD Register (Offset (x8) = 200h, Offset (x16) = 100h) [Reset = 0000001h]

CAN\_IF1CMD is shown in [Figure 16-43](#) and described in [Table 16-31](#).

Return to the [Summary Table](#).

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed. The following points must be borne in mind while writing to this register: (1) Do not write zeros to the whole register. (2) Write to the register in a single 32-bit write or write the upper 16-bits before writing to the lower 16-bits.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 16-43. 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/W-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

**Table 16-31. CAN\_IF1CMD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23	DIR	R/W	0h	Write/Read 0 Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. That is, transfer data from the mailbox into the selected IF1/IF2 Message Buffer Registers. 1 Direction = Write: Transfer direction is from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]) . That is, transfer data from the selected IF1/IF2 Message Buffer Registers to the mailbox. The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 16-31. CAN\_IF1CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	Mask	R/W	0h	<p>Access Mask Bits</p> <p>0 Mask bits will not be changed</p> <p>1 (Direction = Read): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
21	Arb	R/W	0h	<p>Access Arbitration Bits</p> <p>0 Arbitration bits will not be changed</p> <p>1 (Direction = Read): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
20	Control	R/W	0h	<p>Access control bits.</p> <p>If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/NewDat bit in the IF1 message control register will be ignored.</p> <p>0 Control bits will not be changed.</p> <p>1 (Direction = Read): The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1 register set.</p> <p>1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]).</p> <p>Note: This bit is write protected by the Busy bit.</p> <p>Reset type: SYSRSn</p>
19	ClrIntPnd	R/W	0h	<p>Clear Interrupt Pending Bit</p> <p>0 IntPnd bit will not be changed</p> <p>1 (Direction = Read): Clears IntPnd bit in the message object.</p> <p>1 (Direction = Write): This bit is ignored.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
18	TXRQST	R/W	0h	<p>Access Transmission Request (TxRqst) / New Data (NewDat) Bit</p> <p>0 (Direction = Read): NewDat bit will not be changed.</p> <p>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.</p> <p>1 (Direction = Read): Clears NewDat bit in the message object.</p> <p>1 (Direction = Write): Sets TxRqst/NewDat in message object.</p> <p>Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 Message Control Register.</p> <p>Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>

**Table 16-31. CAN\_IF1CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	DATA_A	R/W	0h	Access Data Bytes 0-3 0 Data Bytes 0-3 will not be changed. 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. 1 (Direction = Write): The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
16	DATA_B	R/W	0h	Access Data Bytes 4-7 0 Data Bytes 4-7 will not be changed. 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. 1 (Direction = Write): The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
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. Reset type: SYSRSn
14	RESERVED	R/W	0h	Reserved
13-8	RESERVED	R	0h	Reserved
7-0	MSG_NUM	R/W	1h	Number of message object in Message RAM which is used for data transfer 0x00 Invalid message number 0x01-0x20 Valid message numbers 0x21-0xFF Invalid message numbers Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 16.15.2.23 CAN\_IF1MSK Register (Offset (x8) = 208h, Offset (x16) = 104h) [Reset = FFFFFFFFh]

CAN\_IF1MSK is shown in [Figure 16-44](#) and described in [Table 16-32](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 16-44. CAN\_IF1MSK Register**

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

**Table 16-32. CAN\_IF1MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier 0 The extended identifier bit (Xtd) has no effect on the acceptance filtering. 1 The extended identifier bit (Xtd) is used for acceptance filtering. When 11-bit ('standard') identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	MDir	R/W	1h	Mask Message Direction 0 The message direction bit (Dir) has no effect on the acceptance filtering. 1 The message direction bit (Dir) is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R/W	1FFFFFFFh	Identifier Mask- 0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 16.15.2.24 CAN\_IF1ARB Register (Offset (x8) = 210h, Offset (x16) = 108h) [Reset = 0000000h]

CAN\_IF1ARB is shown in [Figure 16-45](#) and described in [Table 16-33](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 16-45. CAN\_IF1ARB Register**

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

**Table 16-33. CAN\_IF1ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	Message Valid 0 The mailbox is disabled. (The message object is ignored by the message handler). 1 The mailbox is enabled. (The message object is to be used by the message handler). The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets the InIt bit in the CAN Control Register. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	Xtd	R/W	0h	Extended Identifier 0 The 11-bit ('standard') Identifier is used for this message object. 1 The 29-bit ('extended') Identifier is used for this message object. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 16-33. CAN\_IF1ARB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
29	Dir	R/W	0h	Message Direction 0 Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that frame is stored in this message object. 1 Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one). Note: This bit is write protected by Busy bit. Reset type: SYSRSn
28-0	ID	R/W	0h	Message Identifier ID[28:0] 29-bit Identifier ('Extended Frame') ID[28:18] 11-bit Identifier ('Standard Frame') Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 16.15.2.25 CAN\_IF1MCTL Register (Offset (x8) = 218h, Offset (x16) = 10Ch) [Reset = 0000000h]

CAN\_IF1MCTL is shown in [Figure 16-46](#) and described in [Table 16-34](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. This register has control/status bits pertaining to interrupts, acceptance mask, remote frames and FIFO option.

**Figure 16-46. 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			

**Table 16-34. CAN\_IF1MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R/W	0h	New Data 0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1 The message handler or the CPU has written new data into the data portion of this message object. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	MsgLst	R/W	0h	Message Lost (only valid for message objects with direction = receive) 0 No message lost since the last time when this bit was reset by the CPU. 1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
13	IntPnd	R/W	0h	Interrupt Pending 0 This message object is not the source of an interrupt. 1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 16-34. CAN\_IF1MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	TxE	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> <p>Reset type: SYSRSn</p>
10	RxE	R/W	0h	<p>Receive Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful reception of a frame.</p> <p>1 IntPnd will be triggered after the successful reception of a frame.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
9	RmtEn	R/W	0h	<p>Remote Enable</p> <p>0 At the reception of a remote frame, TxRqst is not changed.</p> <p>1 At the reception of a remote frame, TxRqst is set.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
8	TxRqst	R/W	0h	<p>Transmit Request</p> <p>0 This message object is not waiting for a transmission.</p> <p>1 The transmission of this message object is requested and is not yet done.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
7	EoB	R/W	0h	<p>End of Block</p> <p>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.</p> <p>1 The message object is a single message object or the last message object in a FIFO Buffer Block.</p> <p>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R/W	0h	<p>Data length code</p> <p>0-8 Data frame has 0-8 data bytes.</p> <p>9-15 Data frame has 8 data bytes.</p> <p>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>

### 16.15.2.26 CAN\_IF1DATA Register (Offset (x8) = 220h, Offset (x16) = 110h) [Reset = 0000000h]

CAN\_IF1DATA is shown in [Figure 16-47](#) and described in [Table 16-35](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 16-47. CAN\_IF1DATA Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_3								Data_2								Data_1								Data_0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 16-35. CAN\_IF1DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R/W	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R/W	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R/W	0h	Data Byte 0 Reset type: SYSRSn

### 16.15.2.27 CAN\_IF1DATB Register (Offset (x8) = 228h, Offset (x16) = 114h) [Reset = 0000000h]

CAN\_IF1DATB is shown in [Figure 16-48](#) and described in [Table 16-36](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 16-48. CAN\_IF1DATB Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_7								Data_6								Data_5								Data_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 16-36. CAN\_IF1DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R/W	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R/W	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R/W	0h	Data Byte 4 Reset type: SYSRSn

### 16.15.2.28 CAN\_IF2CMD Register (Offset (x8) = 240h, Offset (x16) = 120h) [Reset = 0000001h]

CAN\_IF2CMD is shown in [Figure 16-49](#) and described in [Table 16-37](#).

Return to the [Summary Table](#).

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed. The following points must be borne in mind while writing to this register: (1) Do not write zeros to the whole register. (2) Write to the register in a single 32-bit write or write the upper 16-bits before writing to the lower 16-bits.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 16-49. 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/W-0h	R-0h					
7	6	5	4	3	2	1	0
MSG_NUM							
R/W-1h							

**Table 16-37. CAN\_IF2CMD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23	DIR	R/W	0h	Write/Read 0 Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. That is, transfer data from the mailbox into the selected IF1/IF2 Message Buffer Registers. 1 Direction = Write: Transfer direction is from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]) . That is, transfer data from the selected IF1/IF2 Message Buffer Registers to the mailbox. The other bits of IF1/IF2 Command Mask Register have different functions depending on the transfer direction. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 16-37. CAN\_IF2CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
22	Mask	R/W	0h	<p>Access Mask Bits</p> <p>0 Mask bits will not be changed</p> <p>1 (Direction = Read): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
21	Arb	R/W	0h	<p>Access Arbitration Bits</p> <p>0 Arbitration bits will not be changed</p> <p>1 (Direction = Read): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.</p> <p>1 (Direction = Write): The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
20	Control	R/W	0h	<p>Access control bits.</p> <p>If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/NewDat bit in the IF1 message control register will be ignored.</p> <p>0 Control bits will not be changed.</p> <p>1 (Direction = Read): The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1 register set.</p> <p>1 (Direction = Write): The message control bits will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]).</p> <p>Note: This bit is write protected by the Busy bit.</p> <p>Reset type: SYSRSn</p>
19	ClrIntPnd	R/W	0h	<p>Clear Interrupt Pending Bit</p> <p>0 IntPnd bit will not be changed</p> <p>1 (Direction = Read): Clears IntPnd bit in the message object.</p> <p>1 (Direction = Write): This bit is ignored.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
18	TxRqst	R/W	0h	<p>Access Transmission Request (TxRqst) / New Data (NewDat) Bit</p> <p>0 (Direction = Read): NewDat bit will not be changed.</p> <p>0 (Direction = Write): TxRqst/NewDat bit will be handled according to the Control bit.</p> <p>1 (Direction = Read): Clears NewDat bit in the message object.</p> <p>1 (Direction = Write): Sets TxRqst/NewDat in message object.</p> <p>Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 Message Control Register.</p> <p>Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>

**Table 16-37. CAN\_IF2CMD Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	DATA_A	R/W	0h	Access Data Bytes 0-3 0 Data Bytes 0-3 will not be changed. 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. 1 (Direction = Write): The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
16	DATA_B	R/W	0h	Access Data Bytes 4-7 0 Data Bytes 4-7 will not be changed. 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. 1 (Direction = Write): The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
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. Reset type: SYSRSn
14	RESERVED	R/W	0h	Reserved
13-8	RESERVED	R	0h	Reserved
7-0	MSG_NUM	R/W	1h	Number of message object in Message RAM which is used for data transfer 0x00 Invalid message number 0x01-0x20 Valid message numbers 0x21-0xFF Invalid message numbers Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 16.15.2.29 CAN\_IF2MSK Register (Offset (x8) = 248h, Offset (x16) = 124h) [Reset = FFFFFFFFh]

CAN\_IF2MSK is shown in [Figure 16-50](#) and described in [Table 16-38](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 16-50. CAN\_IF2MSK Register**

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

**Table 16-38. CAN\_IF2MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R/W	1h	Mask Extended Identifier 0 The extended identifier bit (Xtd) has no effect on the acceptance filtering. 1 The extended identifier bit (Xtd) is used for acceptance filtering. When 11-bit ('standard') identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	MDir	R/W	1h	Mask Message Direction 0 The message direction bit (Dir) has no effect on the acceptance filtering. 1 The message direction bit (Dir) is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R/W	1FFFFFFFh	Identifier Mask 0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

### 16.15.2.30 CAN\_IF2ARB Register (Offset (x8) = 250h, Offset (x16) = 128h) [Reset = 0000000h]

CAN\_IF2ARB is shown in [Figure 16-51](#) and described in [Table 16-39](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

Note: While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write-protected.

**Figure 16-51. CAN\_IF2ARB Register**

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

**Table 16-39. CAN\_IF2ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R/W	0h	Message Valid 0 The mailbox is disabled. (The message object is ignored by the message handler). 1 The mailbox is enabled. (The message object is to be used by the message handler). The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets the InIt bit in the CAN Control Register. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
30	Xtd	R/W	0h	Extended Identifier 0 The 11-bit ('standard') Identifier is used for this message object. 1 The 29-bit ('extended') Identifier is used for this message object. Note: This bit is write protected by Busy bit. Reset type: SYSRSn

**Table 16-39. CAN\_IF2ARB Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
29	Dir	R/W	0h	<p>Message Direction</p> <p>0 Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that frame is stored in this message object.</p> <p>1 Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
28-0	ID	R/W	0h	<p>Message Identifier</p> <p>ID[28:0] 29-bit Identifier ('Extended Frame')</p> <p>ID[28:18] 11-bit Identifier ('Standard Frame')</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>

### 16.15.2.31 CAN\_IF2MCTL Register (Offset (x8) = 258h, Offset (x16) = 12Ch) [Reset = 0000000h]

CAN\_IF2MCTL is shown in [Figure 16-52](#) and described in [Table 16-40](#).

Return to the [Summary Table](#).

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. This register has control/status bits pertaining to interrupts, acceptance mask, remote frames and FIFO option.

**Figure 16-52. CAN\_IF2MCTL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
EoB	RESERVED			DLC			
R/W-0h	R-0h			R/W-0h			

**Table 16-40. CAN\_IF2MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R/W	0h	New Data 0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1 The message handler or the CPU has written new data into the data portion of this message object. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
14	MsgLst	R/W	0h	Message Lost (only valid for message objects with direction = receive) 0 No message lost since the last time when this bit was reset by the CPU. 1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
13	IntPnd	R/W	0h	Interrupt Pending 0 This message object is not the source of an interrupt. 1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority. Note: This bit is write protected by Busy bit. Reset type: SYSRSn
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. Reset type: SYSRSn

**Table 16-40. CAN\_IF2MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	TxE	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> <p>Reset type: SYSRSn</p>
10	RxE	R/W	0h	<p>Receive Interrupt Enable</p> <p>0 IntPnd will not be triggered after the successful reception of a frame.</p> <p>1 IntPnd will be triggered after the successful reception of a frame.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
9	RmtEn	R/W	0h	<p>Remote Enable</p> <p>0 At the reception of a remote frame, TxRqst is not changed.</p> <p>1 At the reception of a remote frame, TxRqst is set.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
8	TxRqst	R/W	0h	<p>Transmit Request</p> <p>0 This message object is not waiting for a transmission.</p> <p>1 The transmission of this message object is requested and is not yet done.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
7	EoB	R/W	0h	<p>End of Block</p> <p>0 The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.</p> <p>1 The message object is a single message object or the last message object in a FIFO Buffer Block.</p> <p>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R/W	0h	<p>Data length code</p> <p>0-8 Data frame has 0-8 data bytes.</p> <p>9-15 Data frame has 8 data bytes.</p> <p>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.</p> <p>Note: This bit is write protected by Busy bit.</p> <p>Reset type: SYSRSn</p>

### 16.15.2.32 CAN\_IF2DATA Register (Offset (x8) = 260h, Offset (x16) = 130h) [Reset = 0000000h]

CAN\_IF2DATA is shown in [Figure 16-53](#) and described in [Table 16-41](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 16-53. CAN\_IF2DATA Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_3								Data_2								Data_1								Data_0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 16-41. CAN\_IF2DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R/W	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R/W	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R/W	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R/W	0h	Data Byte 0 Reset type: SYSRSn

### 16.15.2.33 CAN\_IF2DATB Register (Offset (x8) = 268h, Offset (x16) = 134h) [Reset = 0000000h]

CAN\_IF2DATB is shown in [Figure 16-54](#) and described in [Table 16-42](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message. The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order. In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first. All bits in this register are write-protected by the Busy bit.

**Figure 16-54. CAN\_IF2DATB Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_7								Data_6								Data_5								Data_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 16-42. CAN\_IF2DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R/W	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R/W	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R/W	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R/W	0h	Data Byte 4 Reset type: SYSRSn

### 16.15.2.34 CAN\_IF3MSK Register (Offset (x8) = 288h, Offset (x16) = 144h) [Reset = FFFFFFFFh]

CAN\_IF3MSK is shown in [Figure 16-55](#) and described in [Table 16-43](#).

Return to the [Summary Table](#).

This register provides a window to the acceptance mask for the chosen mailbox.

**Figure 16-55. CAN\_IF3MSK Register**

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

**Table 16-43. CAN\_IF3MSK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MXtd	R	1h	Mask Extended Identifier 0 The extended identifier bit (Xtd) has no effect on the acceptance filtering. 1 The extended identifier bit (Xtd) is used for acceptance filtering. Note: When 11-bit ('standard') identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. Reset type: SYSRSn
30	MDir	R	1h	Mask Message Direction 0 The message direction bit (Dir) has no effect on the acceptance filtering. 1 The message direction bit (Dir) is used for acceptance filtering. Reset type: SYSRSn
29	RESERVED	R	1h	Reserved
28-0	Msk	R	1FFFFFFFh	Identifier Mask Identifier Mask 0 The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1 The corresponding bit in the identifier of the message object is used for acceptance filtering. Identifier Mask Reset type: SYSRSn

### 16.15.2.35 CAN\_IF3ARB Register (Offset (x8) = 290h, Offset (x16) = 148h) [Reset = 00000000h]

CAN\_IF3ARB is shown in [Figure 16-56](#) and described in [Table 16-44](#).

Return to the [Summary Table](#).

The bits of the IF3 Arbitration Register mirrors the arbitration bits of a message object.

**Figure 16-56. CAN\_IF3ARB Register**

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

**Table 16-44. CAN\_IF3ARB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	MsgVal	R	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.</p> <p>Reset type: SYSRSn</p>
30	Xtd	R	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>Reset type: SYSRSn</p>
29	Dir	R	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>Reset type: SYSRSn</p>
28-0	ID	R	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>Reset type: SYSRSn</p>

**16.15.2.36 CAN\_IF3MCTL Register (Offset (x8) = 298h, Offset (x16) = 14Ch) [Reset = 0000000h]**

 CAN\_IF3MCTL is shown in [Figure 16-57](#) and described in [Table 16-45](#).

 Return to the [Summary Table](#).

The bits of the IF3 Message Control Register mirrors the message control bits of a message object.

**Figure 16-57. CAN\_IF3MCTL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	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			

**Table 16-45. CAN\_IF3MCTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NewDat	R	0h	New Data 0 No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1 The message handler or the CPU has written new data into the data portion of this message object. Reset type: SYSRSn
14	MsgLst	R	0h	Message Lost (only valid for message objects with direction = receive) 0 No message lost since the last time when this bit was reset by the CPU. 1 The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten. Reset type: SYSRSn
13	IntPnd	R	0h	Interrupt Pending 0 This message object is not the source of an interrupt. 1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority. Reset type: SYSRSn
12	UMask	R	0h	Use Acceptance Mask 0 Mask ignored 1 Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. Reset type: SYSRSn

**Table 16-45. CAN\_IF3MCTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	TxE	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. Reset type: SYSRSn
10	RxE	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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn
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. Reset type: SYSRSn
6-4	RESERVED	R	0h	Reserved
3-0	DLC	R	0h	Data length code 0-8 Data frame has 0-8 data bytes. 9-15 Data frame has 8 data bytes. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. Reset type: SYSRSn

### 16.15.2.37 CAN\_IF3DATA Register (Offset (x8) = 2A0h, Offset (x16) = 150h) [Reset = 0000000h]

CAN\_IF3DATA is shown in [Figure 16-58](#) and described in [Table 16-46](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message.

**Figure 16-58. CAN\_IF3DATA Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_3								Data_2								Data_1								Data_0							
R-0h								R-0h								R-0h								R-0h							

**Table 16-46. CAN\_IF3DATA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_3	R	0h	Data Byte 3 Reset type: SYSRSn
23-16	Data_2	R	0h	Data Byte 2 Reset type: SYSRSn
15-8	Data_1	R	0h	Data Byte 1 Reset type: SYSRSn
7-0	Data_0	R	0h	Data Byte 0 Reset type: SYSRSn

**16.15.2.38 CAN\_IF3DATB Register (Offset (x8) = 2A8h, Offset (x16) = 154h) [Reset = 0000000h]**

CAN\_IF3DATB is shown in [Figure 16-59](#) and described in [Table 16-47](#).

Return to the [Summary Table](#).

This register provides a window to the data bytes of the CAN message.

**Figure 16-59. CAN\_IF3DATB Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data_7								Data_6								Data_5								Data_4							
R-0h								R-0h								R-0h								R-0h							

**Table 16-47. CAN\_IF3DATB Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	Data_7	R	0h	Data Byte 7 Reset type: SYSRSn
23-16	Data_6	R	0h	Data Byte 6 Reset type: SYSRSn
15-8	Data_5	R	0h	Data Byte 5 Reset type: SYSRSn
7-0	Data_4	R	0h	Data Byte 4 Reset type: SYSRSn

### 16.15.2.39 CAN\_IF3UPD Register (Offset (x8) = 2C0h, Offset (x16) = 160h) [Reset = 0000000h]

CAN\_IF3UPD is shown in [Figure 16-60](#) and described in [Table 16-48](#).

Return to the [Summary Table](#).

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. Note: IF3 Update enable should not be set for transmit objects.

**Figure 16-60. CAN\_IF3UPD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IF3UpdEn																															
R/W-0h																															

**Table 16-48. 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. Reset type: SYSRSn

### 16.15.3 CAN Registers to Driverlib Functions

**Table 16-49. CAN Registers to Driverlib Functions**

File	Driverlib Function
<b>CAN_CTL</b>	
can.c	CAN_initModule
can.c	CAN_setBitTiming
can.h	CAN_startModule
can.h	CAN_enableController
can.h	CAN_disableController
can.h	CAN_enableTestMode
can.h	CAN_disableTestMode
can.h	CAN_setInterruptionDebugMode
can.h	CAN_disableAutoBusOn
can.h	CAN_enableAutoBusOn
can.h	CAN_enableInterrupt
can.h	CAN_disableInterrupt
can.h	CAN_enableRetry
can.h	CAN_disableRetry
can.h	CAN_isRetryEnabled
<b>CAN_ES</b>	
can.c	CAN_clearInterruptStatus
can.h	CAN_getStatus
<b>CAN_ERRC</b>	
can.h	CAN_getErrorCount
<b>CAN_BTR</b>	
can.c	CAN_setBitTiming

**Table 16-49. CAN Registers to Driverlib Functions (continued)**

File	Driverlib Function
can.h	CAN_getBitTiming
<b>CAN_INT</b>	
can.h	CAN_getInterruptCause
<b>CAN_TEST</b>	
can.h	CAN_enableTestMode
can.h	CAN_disableTestMode
can.h	CAN_enableMemoryAccessMode
can.h	CAN_disableMemoryAccessMode
<b>CAN_PERR</b>	
-	
<b>CAN_RAM_INIT</b>	
can.h	CAN_initRAM
<b>CAN_GLB_INT_EN</b>	
can.h	CAN_enableGlobalInterrupt
can.h	CAN_disableGlobalInterrupt
<b>CAN_GLB_INT_FLG</b>	
can.h	CAN_getGlobalInterruptStatus
<b>CAN_GLB_INT_CLR</b>	
can.h	CAN_clearGlobalInterruptStatus
<b>CAN_ABOTR</b>	
can.h	CAN_setAutoBusOnTime
<b>CAN_TXRQ_X</b>	
-	
<b>CAN_TXRQ_21</b>	
can.h	CAN_getTxRequests
<b>CAN_NDAT_X</b>	
-	
<b>CAN_NDAT_21</b>	
can.h	CAN_getNewDataFlags
<b>CAN_IPEN_X</b>	
-	
<b>CAN_IPEN_21</b>	
can.h	CAN_getInterruptMessageSource
<b>CAN_MVAL_X</b>	
-	
<b>CAN_MVAL_21</b>	
can.h	CAN_getValidMessageObjects
<b>CAN_IP_MUX21</b>	
can.h	CAN_getInterruptMux
can.h	CAN_setInterruptMux
<b>CAN_IF1CMD</b>	
can.c	CAN_clearInterruptStatus
can.c	CAN_setupMessageObject
can.c	CAN_sendMessage
can.c	CAN_sendMessage_16bit

**Table 16-49. CAN Registers to Driverlib Functions (continued)**

File	Driverlib Function
can.c	CAN_sendMessage_32bit
can.c	CAN_sendMessage_updateDLC
can.c	CAN_sendRemoteRequestMessage
can.c	CAN_transferMessage
can.c	CAN_clearMessage
can.c	CAN_disableMessageObject
can.c	CAN_disableAllMessageObjects
<b>CAN_IF1MSK</b>	
can.c	CAN_setupMessageObject
<b>CAN_IF1ARB</b>	
can.c	CAN_setupMessageObject
can.c	CAN_clearMessage
can.c	CAN_disableMessageObject
can.c	CAN_disableAllMessageObjects
<b>CAN_IF1MCTL</b>	
can.c	CAN_setupMessageObject
can.c	CAN_sendMessage
can.c	CAN_sendMessage_16bit
can.c	CAN_sendMessage_32bit
can.c	CAN_sendMessage_updateDLC
can.c	CAN_sendRemoteRequestMessage
<b>CAN_IF1DATA</b>	
can.c	CAN_sendMessage
can.c	CAN_sendMessage_16bit
can.c	CAN_sendMessage_32bit
can.c	CAN_sendMessage_updateDLC
<b>CAN_IF1DATB</b>	
-	See IF1DATA
<b>CAN_IF2CMD</b>	
can.c	CAN_readMessage
can.c	CAN_transferMessage
<b>CAN_IF2MSK</b>	
-	
<b>CAN_IF2ARB</b>	
can.c	CAN_readMessageWithID
<b>CAN_IF2MCTL</b>	
can.c	CAN_readMessage
<b>CAN_IF2DATA</b>	
can.c	CAN_readMessage
<b>CAN_IF2DATB</b>	
-	See IF2DATA
<b>CAN_IF3MSK</b>	
-	
<b>CAN_IF3ARB</b>	
-	

**Table 16-49. CAN Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>CAN_IF3MCTL</b>	
-	
<b>CAN_IF3DATA</b>	
-	
<b>CAN_IF3DATB</b>	
-	See IF3DATA
<b>CAN_IF3UPD</b>	
-	

This page intentionally left blank.

## Chapter 17 Inter-Integrated Circuit Module (I2C)



This chapter describes the features and operation of the inter-integrated circuit (I2C) module. The I2C module provides an interface between one of these devices and devices compliant with the NXP Semiconductors Inter-IC bus (I2C bus) specification version 2.1, and connected by way of an I2C bus. External components attached to this 2-wire serial bus can transmit/receive 1- to 8-bit data to/from the device through the I2C module. This chapter assumes the reader is familiar with the I2C bus specification.

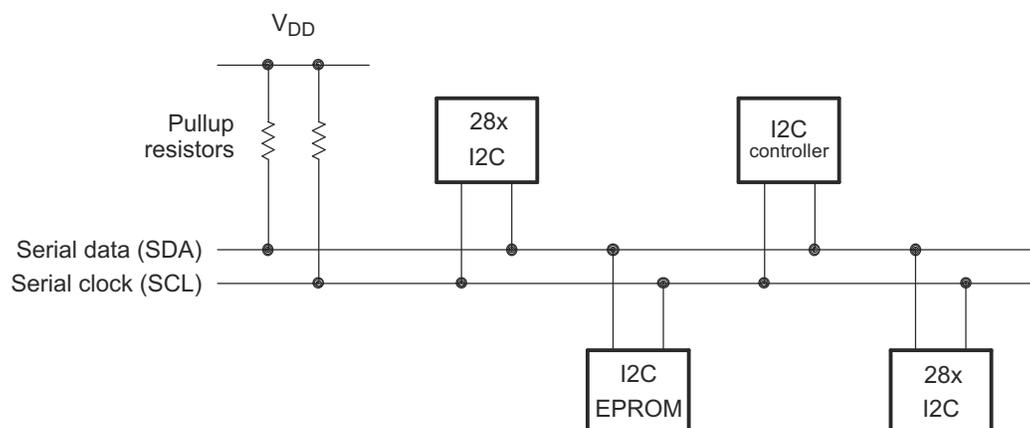
### Note

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 chapter. The number of bits in a data byte is selectable by way of the BC bits of the mode register, I2CMR.

<b>17.1 Introduction</b> .....	<b>1734</b>
<b>17.2 Configuring Device Pins</b> .....	<b>1739</b>
<b>17.3 I2C Module Operational Details</b> .....	<b>1739</b>
<b>17.4 Interrupt Requests Generated by the I2C Module</b> .....	<b>1750</b>
<b>17.5 Resetting or Disabling the I2C Module</b> .....	<b>1754</b>
<b>17.6 Software</b> .....	<b>1755</b>
<b>17.7 I2C Registers</b> .....	<b>1757</b>

## 17.1 Introduction

The I2C module supports any slave or master I2C-compatible device. [Figure 17-1](#) shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.



**Figure 17-1. Multiple I2C Modules Connected**

### 17.1.1 I2C Related Collateral

#### Foundational Materials

- [C2000 Academy - I2C](#)
- [I2C Hardware Overview \(Video\)](#)
- [I2C Protocol Overview \(Video\)](#)
- [Understanding the I2C Bus Application Report](#)

#### Getting Started Materials

- [Configuring the TMS320F280x DSP as an I2C Processor Application Report](#)
- [I2C Buffers Overview \(Video\)](#)
- [I2C Dynamic Addressing Application Report](#)
- [I2C translators overview \(Video\)](#)
- [Interfacing EEPROM Using C2000 I2C Module Application Report](#)
- [Why, When, and How to use I2C Buffers Application Report](#)

#### Expert Materials

- [I2C Bus Pull-Up Resistor Calculation Application Report](#)
- [Maximum Clock Frequency of I2C Bus Using Repeaters Application Report](#)

### 17.1.2 Features

The I2C module has the following features:

- Compliance with the NXP Semiconductors I2C bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate from 10kbps up to 400kbps (Fast-mode)
- Receive FIFO and Transmitter FIFO (16-deep x 8-bit FIFO)
- Supports two ePIE interrupts:
  - I2Cx Interrupt – Any of the following events can be configured to generate an I2Cx interrupt:
    - Transmit-data ready
    - Receive-data ready
    - Register-access ready
    - No-acknowledgment received
    - Arbitration lost
    - Stop condition detected
    - Addressed as slave
  - I2Cx\_FIFO interrupts:
    - Transmit FIFO interrupt
    - Receive FIFO interrupt
- Module enable and disable capability
- Free data format mode

### 17.1.3 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode

### 17.1.4 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 [Figure 17-2](#). These two pins carry information between the C28x device and other devices connected to the I2C bus. The SDA and SCL pins are both bidirectional and 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 I2CCNT register in the *I2C Registers* 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 the I2CMDR register in the *I2C Registers* section 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 the I2C module 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 17-2](#) 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.

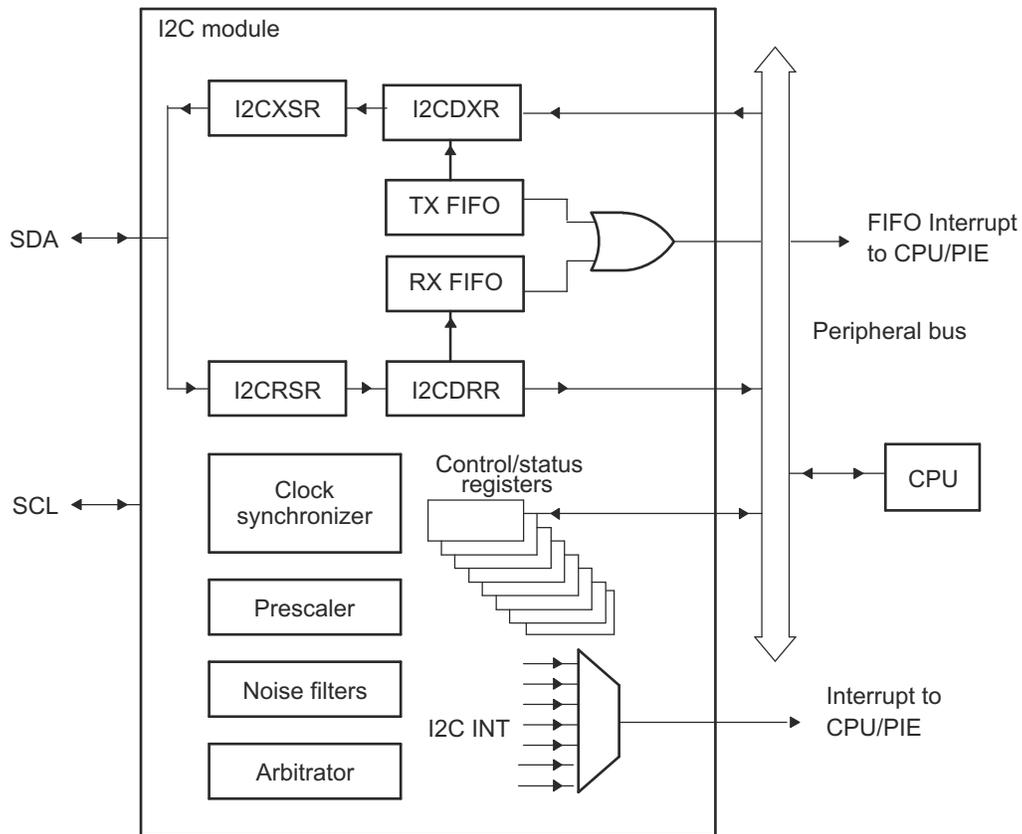


Figure 17-2. I2C Module Conceptual Block Diagram

### 17.1.5 Clock Generation

The I2C module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the SYSCLK to produce the I2C module clock and this I2C module clock is divided further to produce the I2C master clock on the SCL pin. Figure 17-3 shows the clock generation diagram for I2C module.

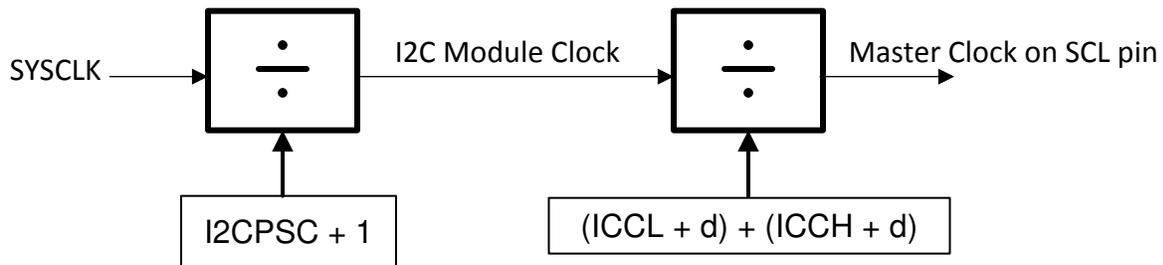


Figure 17-3. Clocking Diagram for the I2C Module

**Note**

To meet all of the I2C protocol timing specifications, the I2C module clock must be between 7 to 12MHz.

To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{I2C Module Clock (Fmod)} = \frac{\text{SYSCLK}}{(\text{I2CPSC} + 1)} \quad (15)$$

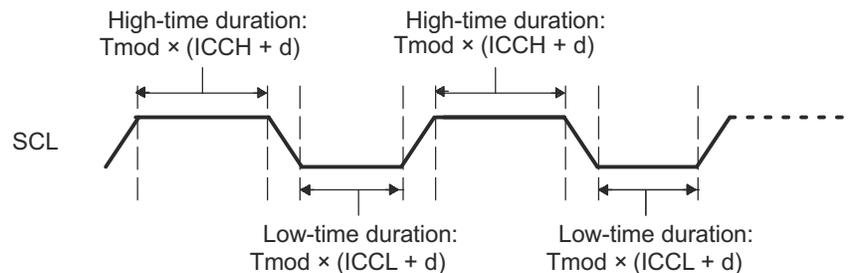
The prescaler must be initialized only while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

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 17-3, 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 17.1.6 for the master clock frequency equation.

### 17.1.6 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in Section 17.1.5, when the I2C module is a master, the I2C module clock is divided down further to use as the master clock on the SCL pin. As shown in Figure 17-4, 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 17-4. Roles of the Clock Divide-Down Values (ICCL and ICCH)**

#### 17.1.6.1 Formula for the Master Clock Period

The master clock period ( $T_{\text{mst}}$ ) is a multiple of the period of the I2C Module Clock ( $T_{\text{mod}}$ ):

$$\text{Master Clock period (Tmst)} = \frac{[(\text{ICCH} + d) + (\text{ICCL} + d)]}{\text{I2C Module Clock (Fmod)}} \quad (16)$$

where  $d$  depends on the divide-down value IPSC, as shown in Table 17-1. IPSC is described in the I2CPSC register.

**Table 17-1. Dependency of Delay  $d$  on the Divide-Down Value IPSC**

IPSC	$d$
0	7
1	6
Greater than 1	5

## 17.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification must 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 *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux and settings.

## 17.3 I2C Module Operational Details

This section provides an overview of the I2C bus protocol and how it is implemented.

### 17.3.1 Input and Output Voltage Levels

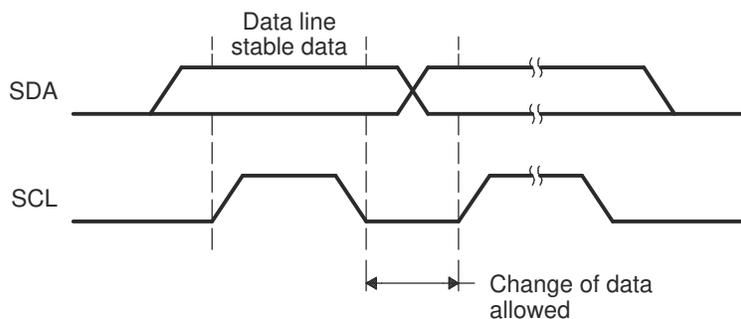
One clock pulse is generated by the master device for each data bit transferred. Due to a variety of different technology devices that can be connected to the I2C bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of  $V_{DD}$ . For details, see the device data sheet.

### 17.3.2 Selecting Pullup Resistors

The chosen pullup resistor must meet the I2C standard timings. In most circumstances, 2.2k $\Omega$  of total bus resistance to VDDIO is sufficient. The value of the pullup resistance used on both the SCL and SDA pins be matched is also recommended. For evaluating pullup resistor values for a particular design, see the [I2C Bus Pullup Resistor Calculation Application Report](#).

### 17.3.3 Data Validity

The data on SDA must be stable during the high period of the clock (see [Figure 17-5](#)). The high or low state of the data line, SDA, must change only when the clock signal on SCL is low.



**Figure 17-5. Bit Transfer on the I2C bus**

### 17.3.4 Operating Modes

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See [Table 17-2](#) for the names and descriptions of the modes.

If the I2C module is a master, the I2C module 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, the I2C module begins as a slave-receiver and typically sends acknowledgment when the I2C module recognizes the slave address from a master. If the master is 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 17-2. Operating Modes of the I2C Module**

Operating Mode	Description
Slave-receiver mode	The I2C module is a slave and receives data from a master.  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 can hold SCL low while the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received. See <a href="#">Section 17.3.8</a> for more details.
Slave-transmitter mode	The I2C module is a slave and transmits data to a master.  This mode can be entered only from the slave-receiver mode; the I2C module must first receive a command from the master. When using any of the 7-bit/10-bit addressing formats, the I2C module enters the slave-transmitter mode if the slave address byte is the same as the address (in I2COAR) and the master has transmitted R/ $\bar{W}$ = 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 <a href="#">Section 17.3.8</a> for more details.
Master-receiver mode	The I2C module is a master and receives data from a slave.  This mode can be entered only from the master-transmitter mode; the I2C module must first transmit a command to the slave. When using any of the 7-bit/10-bit addressing formats, the I2C module enters the master-receiver mode after transmitting the slave address byte and R/ $\bar{W}$ = 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.
Master-transmitter mode	The I2C module is a master and transmits control information and data to a slave.  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.

To summarize, SCL is held low in the following conditions:

- When an overrun condition is detected (RSFULL = 1), in Slave-receiver mode.
- When an underflow condition is detected (XSMT = 0), in Slave-transmitter mode.

I2C slave nodes accept and provide data when the I2C master node requests data.

- To release SCL in slave-receiver mode, read data from I2CDRR.
- To release SCL in slave-transmitter mode, write data to I2CDXR.
- To force a release without handling the data, reset the module using the I2CMDR.IRS bit.

**Table 17-3. Master-Transmitter/Receiver Bus Activity Defined by the RM, STT, and STP Bits of I2CMDR**

RM	STT	STP	Bus Activity <sup>(1)</sup>	Description
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D.	START condition, slave address, n data bytes (n = value in I2CCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, n data bytes, STOP condition (n = value in I2CCNT)
1	0	0	None	No activity
1	0	1	P	STOP condition
1	1	0	S-A-D-D-D.	Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition
1	1	1	None	Reserved bit combination (No activity)

(1) S = START condition; A = Address; D = Data byte; P = STOP condition;

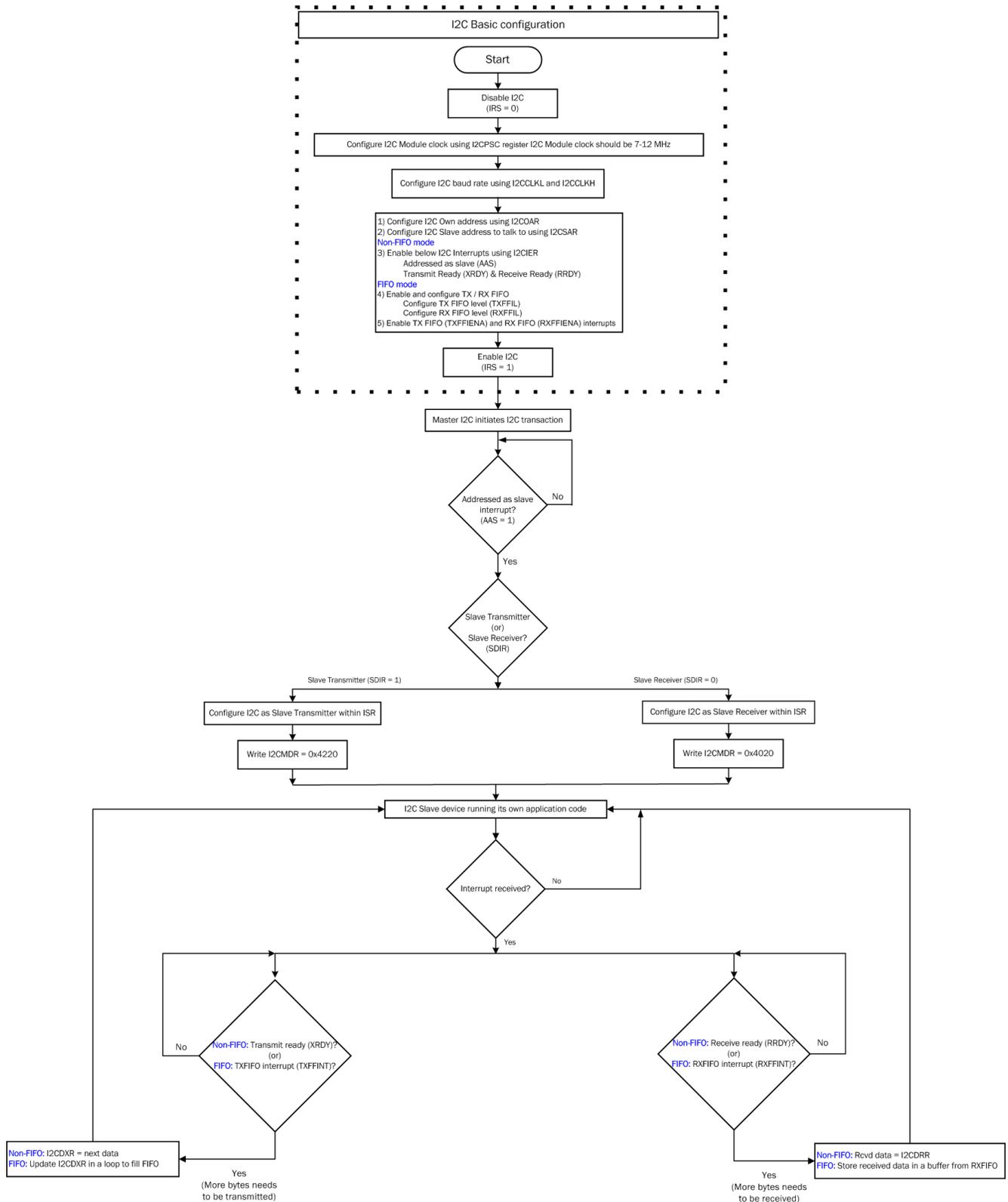


Figure 17-6. I2C Slave TX / RX Flowchart

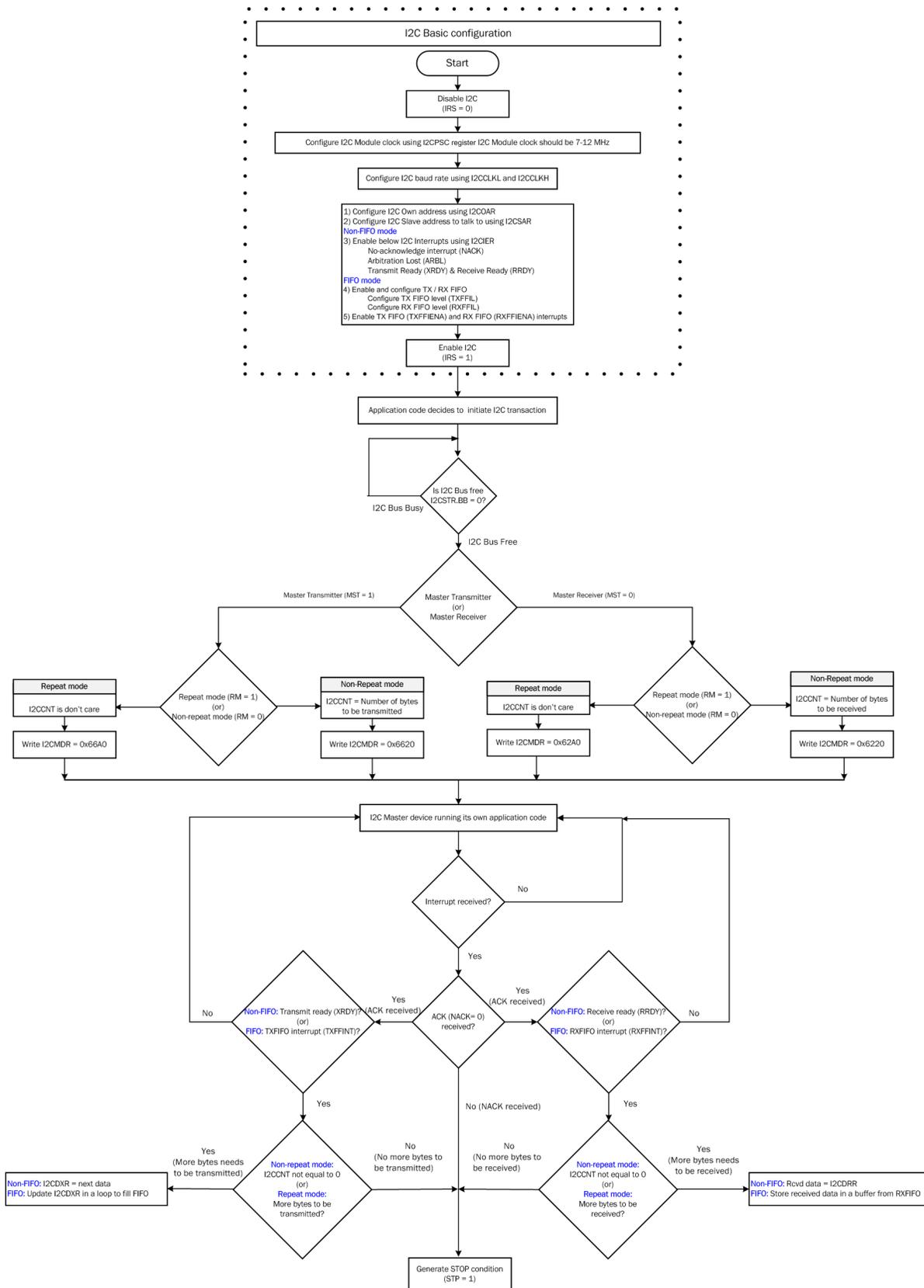
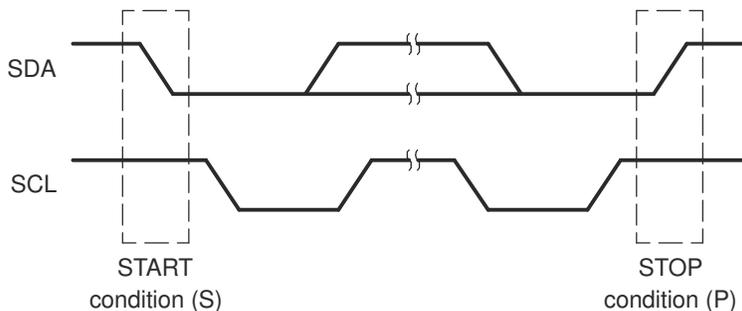


Figure 17-7. I2C Master TX / RX Flowchart

### 17.3.5 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 17-8](#):

- 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 17-8. 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 the bits (including MST, STT, and STP), see [Section 17.7](#).

The I2C peripheral cannot detect a START or STOP condition while in reset (IRS = 0). The BB bit remains in the cleared state (BB = 0) while the I2C peripheral is in reset (IRS = 0). When the I2C peripheral is taken out of reset (IRS set to 1), the BB bit does not correctly reflect the I2C bus status until a START or STOP condition is detected.

Follow these steps before initiating the first data transfer with I2C:

1. After taking the I2C peripheral out of reset by setting the IRS bit to 1, wait a period larger than the total time taken for the longest data transfer in the application. By waiting for a period of time after I2C comes out of reset, make sure that at least one START or STOP condition has occurred on the I2C bus and has been captured by the BB bit. After this period, the BB bit correctly reflects the state of the I2C bus.
2. Check the BB bit and verify that BB = 0 (bus not busy) before proceeding.
3. Begin data transfers.

Not resetting the I2C peripheral in between transfers makes sure that the BB bit reflects the actual bus status. If users must reset the I2C peripheral in between transfers, repeat steps 1 through 3 every time the I2C peripheral is taken out of reset.

### 17.3.6 Non-repeat Mode versus Repeat Mode

#### Non-repeat mode:

- When I2CMDR.RM = 0, I2C module is configured in non-repeat mode.
- I2CCNT register determines the number of bytes to be transmitted or received.
- If STP = 0 in I2CMDR, the ARDY bit is set when the internal data counter counts down to 0.
- If STP = 1, ARDY bit does not get set and I2C module generates a STOP condition when the internal data counter counts down to 0.

---

#### Note

In non-repeat mode (RM = 0), if I2CCNT is set to 0, I2C state machine expects to transmit or receive 65536 bytes and not 0 bytes.

---

#### Repeat mode:

- When I2CMDR.RM = 1, I2C module is configured in repeat mode.
- I2CCNT register contents do not determine the number of bytes to be transmitted or received.
- Number of bytes to be transmitted or received can be controlled by software.
- ARDY bit gets set at end of transmission and reception of each byte.

---

#### Note

Once you start I2C transaction in non-repeat mode or repeat mode, you cannot switch into another mode until the I2C transaction is completed with a STOP condition.

---

### 17.3.7 Serial Data Formats

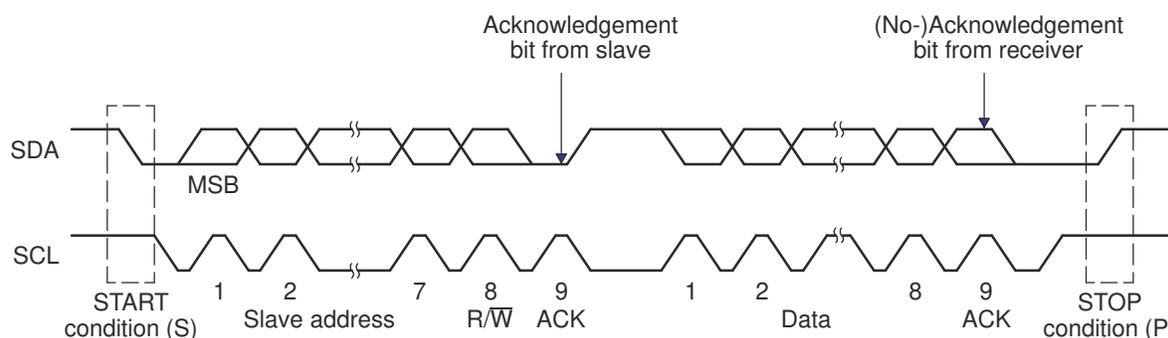
Figure 17-9 shows an example of a data transfer on the I2C bus. The I2C module supports 1 to 8-bit data values. In Figure 17-9, 8-bit data is transferred. Each bit put on the SDA line equates to 1 pulse on the SCL line, and the values are always transferred with the most significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted. The serial data format used in Figure 17-9 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 17-10 through Figure 17-12 and described in the paragraphs that follow the figures.

---

#### Note

In Figure 17-9 through Figure 17-12, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

---



**Figure 17-9. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)**

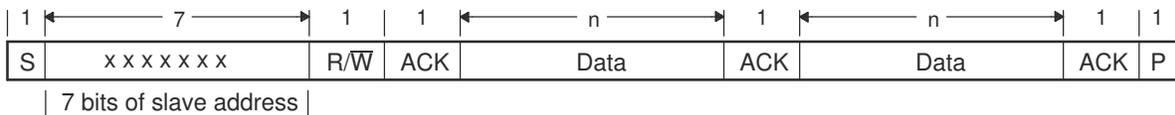
### 17.3.7.1 7-Bit Addressing Format

The 7-bit addressing format is the default format after reset. Disabling expanded address (I2CMDR.XA = 0) and free data format (I2CMDR.FDF = 0) enables 7-bit addressing format.

In this format (see [Figure 17-10](#)), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/W bit. R/W determines the direction of the data:

- R/W = 0: The I2C master writes (transmits) data to the addressed slave. This can be achieved by setting I2CMDR.TRX = 1 (Transmitter mode)
- R/W = 1: The I2C master reads (receives) data from the slave. This can be achieved by setting I2CMDR.TRX = 0 (Receiver mode)

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.

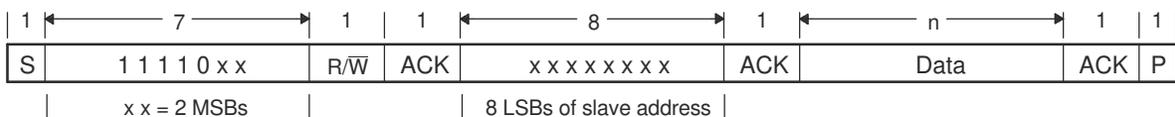


**Figure 17-10. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)**

### 17.3.7.2 10-Bit Addressing Format

The 10-bit addressing format can be enabled by setting expanded address (I2CMDR.XA = 1) and disabling free data format (I2CMDR.FDF = 0).

The 10-bit addressing format (see [Figure 17-11](#)) is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/W. The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. For more details about using 10-bit addressing, see the NXP Semiconductors I2C bus specification.

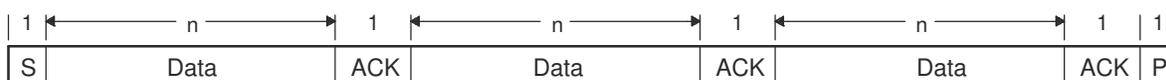


**Figure 17-11. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)**

### 17.3.7.3 Free Data Format

The free data format can be enabled by setting I2CMDR. FDF = 1.

In this format (see [Figure 17-12](#)), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.



**Figure 17-12. I2C Module Free Data Format (FDF = 1 in I2CMDR)**

#### Note

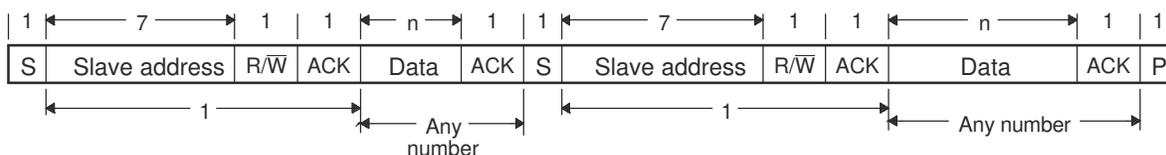
The free data format is not supported in the digital loopback mode (I2CMDR.DLB = 1).

**Table 17-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR**

MST	FDF	I2C Module State	Function of TRX
0	0	In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C module responds as a receiver or a transmitter.
0	1	In slave mode and free data format mode	The free data format mode requires that the I2C module remains the transmitter or the receiver throughout the transfer. TRX identifies the role of the I2C module: TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	0	In master mode but not free data format mode	TRX = 1: The I2C module is a transmitter. TRX = 0: The I2C module is a receiver.
1	1	In master mode and free data format mode	TRX = 0: The I2C module is a receiver. TRX = 1: The I2C module is a transmitter.

### 17.3.7.4 Using a Repeated START Condition

I2C master can communicate with multiple slave addresses without having to give up control of the I2C bus by driving a STOP condition. This can be achieved by driving another START condition at the end of each data type. The repeated START condition can be used with the 7-bit addressing and 10-bit addressing. [Figure 17-13](#) shows a repeated START condition in the 7-bit addressing format.



**Figure 17-13. Repeated START Condition (in This Case, 7-Bit Addressing Format)**

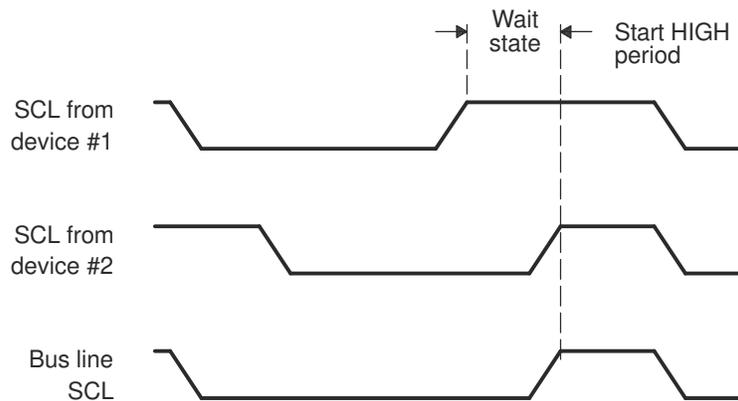
#### Note

In [Figure 17-13](#), n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

### 17.3.8 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 17-14 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 a low period. The SCL is held low by the device with the longest low period. The other devices that finish the low periods must wait for SCL to be released, before starting the 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 17-14. Synchronization of Two I2C Clock Generators During Arbitration**

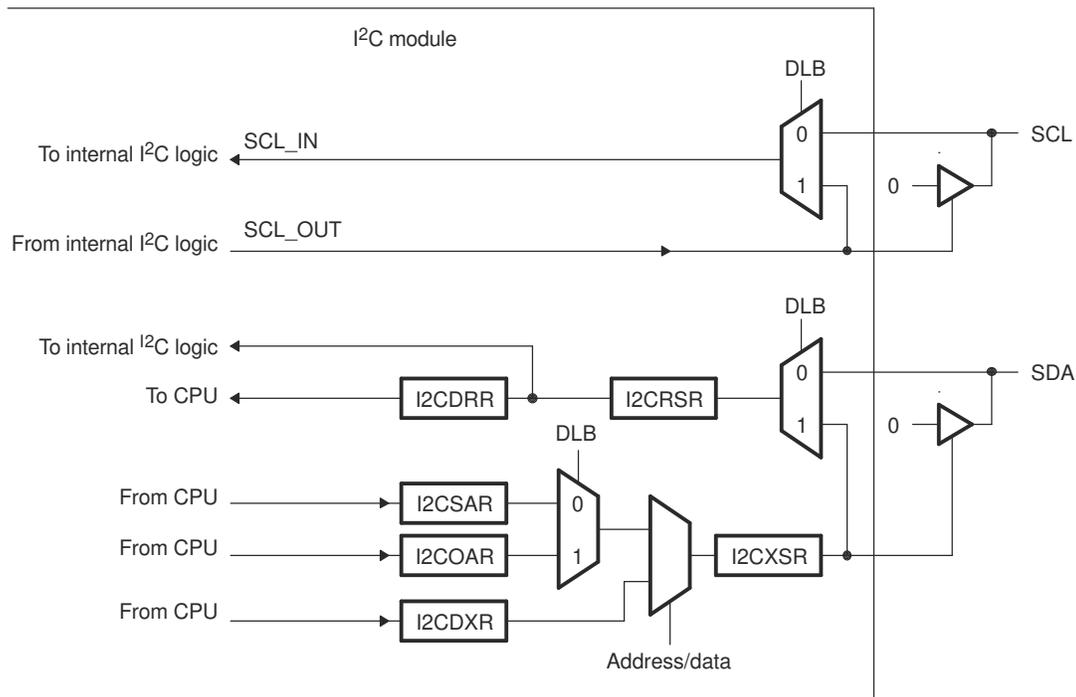


### 17.3.10 Digital Loopback Mode

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:

$$n = 8 * (\text{SYSCLK}) / (\text{I2C module clock (Fmod)})$$

The transmit clock and the receive clock are the same. The address seen on the external SDA pin is the address in the I2COAR register. Figure 17-16 shows the signal routing in digital loopback mode.



**Figure 17-16. 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.

### 17.3.11 NACK Bit Generation

When the I2C module is a receiver (master or slave), the I2C module 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 17-5](#) summarizes the various ways you can allow the I2C module to send a NACK bit.

#### Note

When a NACK is sent, the following occurs:

1. The STP in I2CMR is cleared
2. SCL is held low
3. The NACK in I2CSTR is set

**Table 17-5. Ways to Generate a NACK Bit**

I2C Module Condition	NACK Bit Generation Options
Slave-receiver modes	Allow an overrun condition (RSFULL = 1 in I2CSTR) Reset the module (IRS = 0 in I2CMR) Set the NACKMOD bit of I2CMR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Repeat mode (RM = 1 in I2CMR)	Generate a STOP condition (STP = 1 in I2CMR) Reset the module (IRS = 0 in I2CMR) Set the NACKMOD bit of I2CMR before the rising edge of the last data bit you intend to receive
Master-receiver mode AND Nonrepeat mode (RM = 0 in I2CMR)	If STP = 1 in I2CMR, 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 I2CMR) Set STP = 1 to generate a STOP condition Set the NACKMOD bit of I2CMR before the rising edge of the last data bit you intend to receive

### 17.4 Interrupt Requests Generated by the I2C Module

Each I2C module can generate two CPU interrupts.

1. Basic I2C interrupt: Possible basic I2C interrupt sources that can trigger this interrupt are described in [Section 17.4.1](#).
2. I2C FIFO interrupt: Possible I2C FIFO interrupt sources that can trigger this interrupt are described in [Section 17.4.2](#)

### 17.4.1 Basic I2C Interrupt Requests

The I2C module generates the interrupt requests described in [Table 17-6](#). As shown in [Figure 17-17](#), 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, the 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.

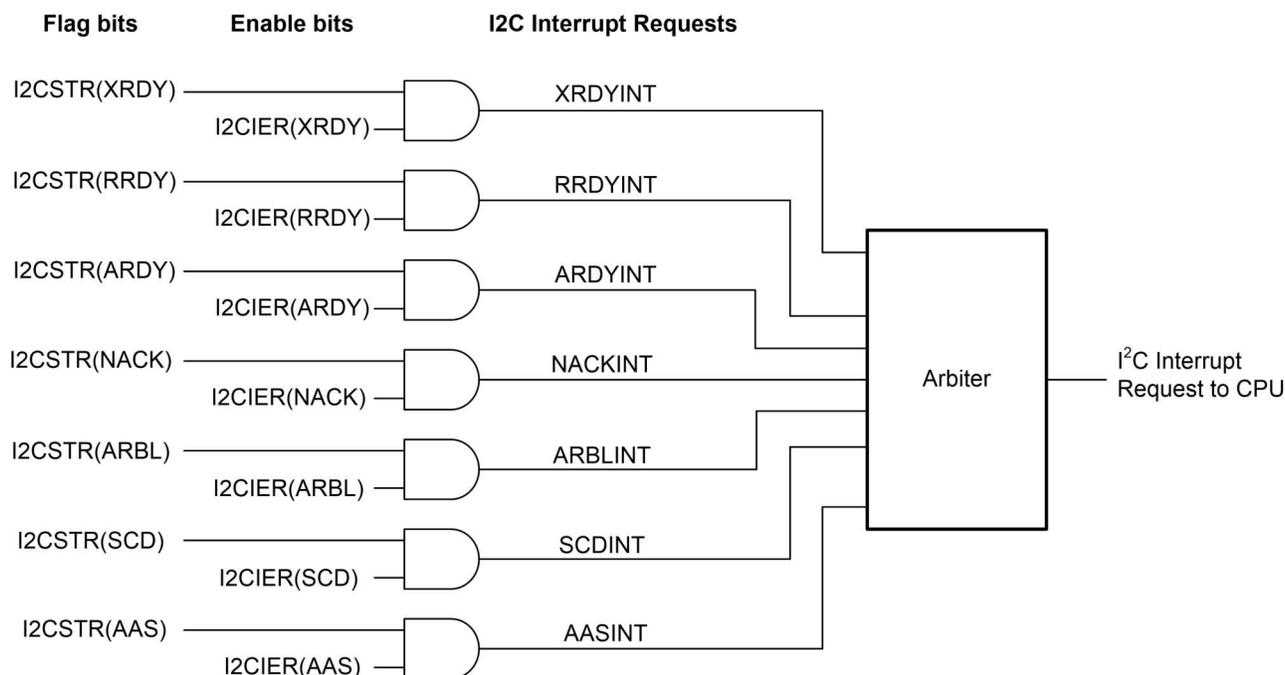
The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if the request is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A\_ISR). The I2CINT1A\_ISR for the I2C interrupt can determine the interrupt source 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 the bit.
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 17-6. Descriptions of the Basic I2C Interrupt Requests**

I2C Interrupt Request	Interrupt Source
XRDYINT	<p>Transmit ready condition: The data transmit register (I2CDXR) is ready to accept new data because the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR).</p> <p>As an alternative to using XRDYINT, the CPU can poll the XRDY bit of the status register, I2CSTR. XRDYINT must not be used when in FIFO mode. Use the FIFO interrupts instead.</p>
RRDYINT	<p>Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR.</p> <p>As an alternative to using RRDYINT, the CPU can poll the RRDY bit of I2CSTR. RRDYINT must not be used when in FIFO mode. Use the FIFO interrupts instead.</p>
ARDYINT	<p>Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used.</p> <p>The specific events that generate ARDYINT are the same events that set the ARDY bit of I2CSTR.</p> <p>As an alternative to using ARDYINT, the CPU can poll the ARDY bit.</p>
NACKINT	<p>No-acknowledgment condition: The I2C module is configured as a master-transmitter and did not received acknowledgment from the slave-receiver.</p> <p>As an alternative to using NACKINT, the CPU can poll the NACK bit of I2CSTR.</p>
ARBLINT	<p>Arbitration-lost condition: The I2C module has lost an arbitration contest with another master-transmitter.</p> <p>As an alternative to using ARBLINT, the CPU can poll the ARBL bit of I2CSTR.</p>
SCDINT	<p>Stop condition detected: A STOP condition was detected on the I2C bus.</p> <p>As an alternative to using SCDINT, the CPU can poll the SCD bit of the status register, I2CSTR.</p>
AASINT	<p>Addressed as slave condition: The I2C has been addressed as a slave device by another master on the I2C bus.</p> <p>As an alternative to using AASINT, the CPU can poll the AAS bit of the status register, I2CSTR.</p>



**Figure 17-17. Enable Paths of the I2C Interrupt Requests**

The priorities of the basic I2C interrupt requests are listed in order of highest priority to lowest priority:

1. ARBLINT
2. NACKINT
3. ARDYINT
4. RRDYINT
5. XRDYINT
6. SCDINT
7. AASINT

The normal transmit interrupt timing makes it possible for stale data to remain in the transmit buffer if a transaction is aborted in the middle of a byte. To avoid this, set the FCM bit in the I2CEMDR register. When this bit is set, the transmit data ready interrupt is generated only when data is required for a bus transaction. In master mode, the interrupt is first generated when the ACK of the address byte is received. In slave mode, the interrupt is first generated when the address is matched. Further interrupts are generated when the data is ACKed. In this mode, XRDY is asserted at the same time as the transmit ready interrupt.

The I2C module has a backwards compatibility bit (BC) in the I2CEMDR register. The timing diagram in [Figure 17-18](#) demonstrates the effect the backwards compatibility bit has on I2C module registers and interrupts when configured as a slave-transmitter.

Slave Transmitter

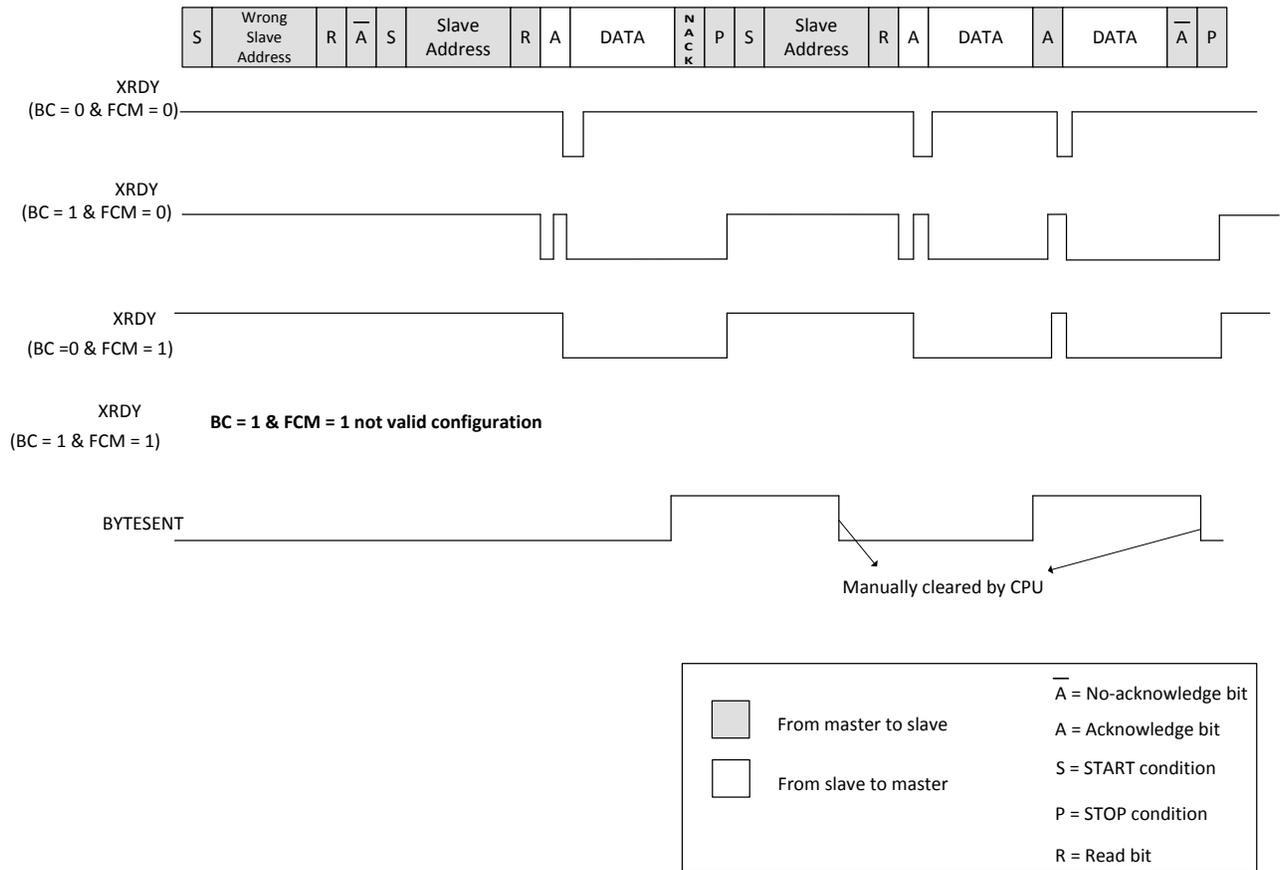


Figure 17-18. Backwards Compatibility Mode and Forward Compatibility Bit, Slave Transmitter

## 17.4.2 I2C FIFO Interrupts

In addition to the seven basic I2C interrupts, the transmit and receive FIFOs each contain the ability to generate an interrupt (I2CINT2A). The transmit FIFO can be configured to generate an interrupt after transmitting a defined number of bytes, up to 16. The receive FIFO can be configured to generate an interrupt after receiving a defined number of bytes, up to 16. These two interrupt sources are ORed together into a single maskable CPU interrupt. Figure 17-19 shows the structure of I2C FIFO interrupt. The interrupt service routine can then read the FIFO interrupt status flags to determine from which source the interrupt came. See the I2C transmit FIFO register (I2CFFTX) and the I2C receive FIFO register (I2CFFRX) descriptions.

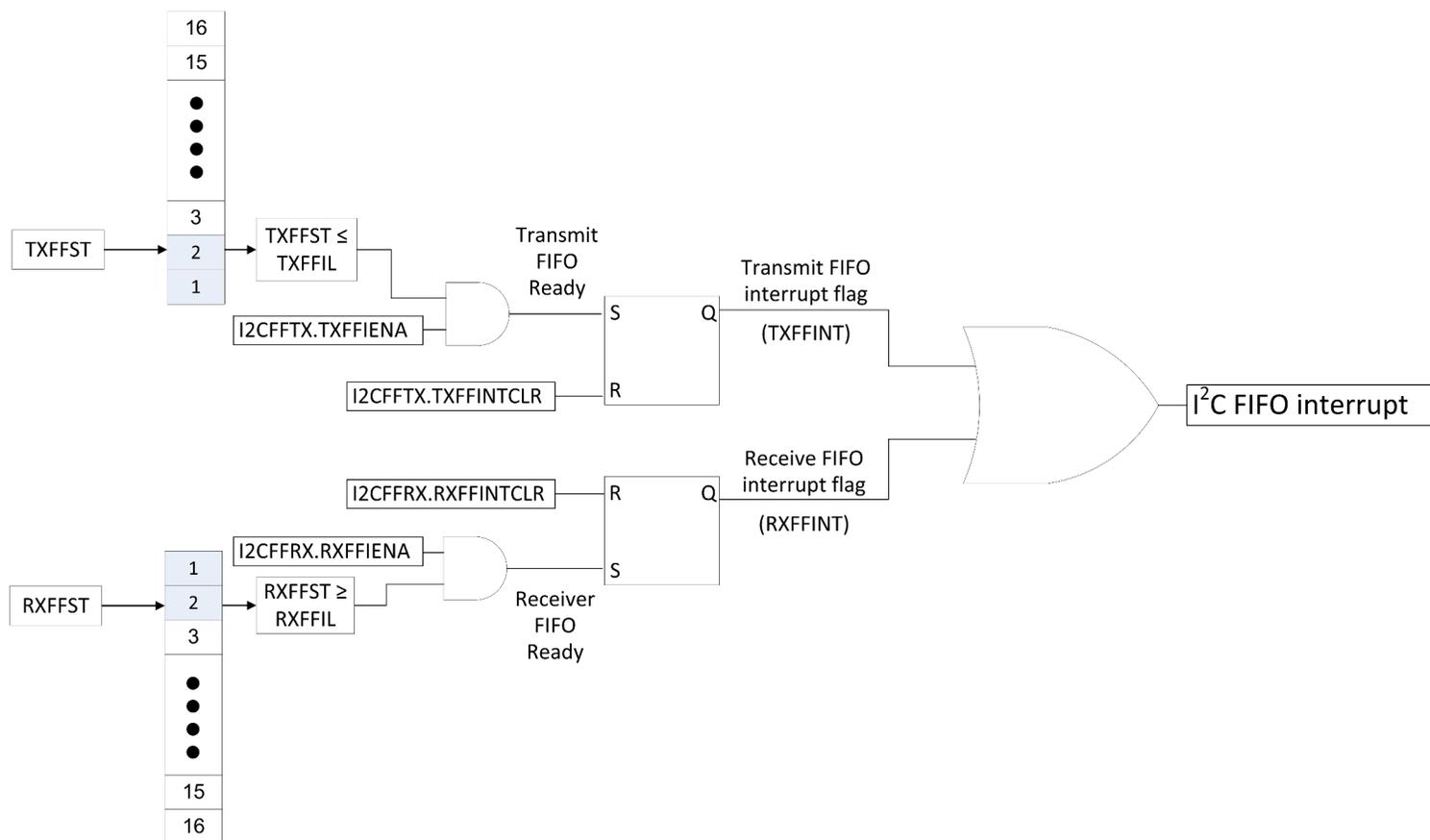


Figure 17-19. I2C FIFO Interrupt

## 17.5 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 (I2CMODR). All status bits (in I2CSTR) are forced to the 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 the 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.

## 17.6 Software

### 17.6.1 I2C Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
 C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/i2c

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 17.6.1.1 C28x-I2C Library source file for FIFO interrupts

FILE: i2cLib\_FIFO\_controller\_interrupt.c

#### 17.6.1.2 C28x-I2C Library source file for FIFO using polling

FILE: i2cLib\_FIFO\_polling.c

#### 17.6.1.3 C28x-I2C Library source file for FIFO interrupts

FILE: i2cLib\_FIFO\_controller\_target\_interrupt.c

#### 17.6.1.4 I2C Digital Loopback with FIFO Interrupts

FILE: i2c\_ex1\_loopback.c

This program uses the internal loopback test mode of the I2C module. Both the TX and RX I2C FIFOs and their interrupts are used. The pinmux and I2C initialization is done through the sysconfig file.

A stream of data is sent and then compared to the received stream. The sent data looks like this:

```
0000 0001
0001 0002
0002 0003
....
00FE 00FF
00FF 0000
etc..
```

This pattern is repeated forever.

#### External Connections

- None

#### Watch Variables

- *sData* - Data to send
- *rData* - Received data
- *rDataPoint* - Used to keep track of the last position in the receive stream for error checking

#### 17.6.1.5 I2C EEPROM

FILE: i2c\_ex2\_eeprom.c

This program will write 1-14 words to EEPROM and read them back. The data written and the EEPROM address written to are contained in the message structure, *i2cMsgOut*. The data read back will be contained in the message structure *i2cMsgIn*.

#### External Connections

- Connect external I2C EEPROM at address 0x50
- Connect `DEVICE_GPIO_PIN_SDAA` on to external EEPROM SDA (serial data) pin
- Connect `DEVICE_GPIO_PIN_SCLA` on to external EEPROM SCL (serial clock) pin

#### Watch Variables

- *i2cMsgOut* - Message containing data to write to EEPROM
- *i2cMsgIn* - Message containing data read from EEPROM

### 17.6.1.6 I2C Digital External Loopback with FIFO Interrupts

FILE: i2c\_ex3\_external\_loopback.c

This program uses the I2CA and I2CB modules for achieving external loopback. The I2CA TX FIFO and the I2CB RX FIFO are used along with their interrupts.

A stream of data is sent on I2CA and then compared to the received stream on I2CB. The sent data looks like this:

```
0000 0001
0001 0002
0002 0003
....
00FE 00FF
00FF 0000
etc..
```

This pattern is repeated forever.

#### External Connections

- Connect SCLA (DEVICE\_GPIO\_PIN\_SCLA) to SCLB (DEVICE\_GPIO\_PIN\_SCLB)
- and SDAA (DEVICE\_GPIO\_PIN\_SDAA) to SDAB (DEVICE\_GPIO\_PIN\_SDAB)
- Connect DEVICE\_GPIO\_PIN\_LED1 to an LED used to depict data transfers.

#### Watch Variables

- *sData* - Data to send
- *rData* - Received data
- *rDataPoint* - Used to keep track of the last position in the receive stream for error checking

### 17.6.1.7 I2C EEPROM

FILE: i2c\_ex4\_eeprom\_polling.c

This program will show how to perform different EEPROM write and read commands using I2C polling method. EEPROM used for this example is AT24C256.

#### External Connections

- Connect external I2C EEPROM at address 0x50 Signal | I2CA | EEPROM

SCL | DEVICE\_GPIO\_PIN\_SCLA | SCL SDA | DEVICE\_GPIO\_PIN\_SDAA | SDA

Make sure to connect GND pins if EEPROM and C2000 device are in different board.

### 17.6.1.8 I2C controller target communication using FIFO interrupts

FILE: i2c\_ex5\_controller\_target\_interrupt.c

This program shows how to use I2CA and I2CB modules in both controller and target configuration. This example uses I2C FIFO interrupts and doesn't use polling.

Example1: I2CA as controller Transmitter and I2CB working target Receiver  
 Example2: I2CA as controller Receiver and I2CB working target Transmitter  
 Example3: I2CB as controller Transmitter and I2CA working target Receiver  
 Example4: I2CB as controller Receiver and I2CA working target Transmitter

External Connections on launchpad should be made as shown below

```
Signal | I2CA | I2CB
SCL | DEVICE_GPIO_PIN_SCLA | DEVICE_GPIO_PIN_SCLB
SDA | DEVICE_GPIO_PIN_SDAA | DEVICE_GPIO_PIN_SDAB
```

Watch Variables in memory window

- *I2CA\_TXdata*
- *I2CA\_RXdata*

- *I2CB\_TXdata*
- *I2CB\_RXdata* stream for error checking

### 17.6.1.9 I2C EEPROM

FILE: i2c\_ex6\_eeprom\_interrupt.c

This program will shows how to perform different EEPROM write and read commands using I2C interrupts  
EEPROM used for this example is AT24C256

#### External Connections

- Connect external I2C EEPROM at address 0x50 Signal | I2CA | EEPROM

SCL | DEVICE\_GPIO\_PIN\_SCL | SCL SDA | DEVICE\_GPIO\_PIN\_SDA | SDA

Make sure to connect GND pins if EEPROM and C2000 device are in different board.

//Example 1: EEPROM Byte Write //Example 2: EEPROM Byte Read //Example 3: EEPROM word (16-bit) write //Example 4: EEPROM word (16-bit) read //Example 5: EEPROM Page write //Example 6: EEPROM word Paged read

#### Watch Variables

- *TX\_MsgBuffer* - Message buffer which stores the data to be transmitted
- *RX\_MsgBuffer* - Message buffer which stores the data to be received

## 17.7 I2C Registers

This section describes the C28x I2C Module Registers.

### 17.7.1 I2C Base Address Table

**Table 17-7. I2C Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
I2caRegs	I2C_REGS	I2CA_BASE	0x0000_7300	YES
I2cbRegs	I2C_REGS	I2CB_BASE	0x0000_7340	YES

## 17.7.2 I2C\_REGS Registers

Table 17-8 lists the memory-mapped registers for the I2C\_REGS registers. 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. I2C\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	I2COAR	I2C Own address		<a href="#">Go</a>
1h	I2CIER	I2C Interrupt Enable		<a href="#">Go</a>
2h	I2CSTR	I2C Status		<a href="#">Go</a>
3h	I2CCLKL	I2C Clock low-time divider		<a href="#">Go</a>
4h	I2CCLKH	I2C Clock high-time divider		<a href="#">Go</a>
5h	I2CCNT	I2C Data count		<a href="#">Go</a>
6h	I2CDRR	I2C Data receive		<a href="#">Go</a>
7h	I2CSAR	I2C Slave address		<a href="#">Go</a>
8h	I2CDXR	I2C Data Transmit		<a href="#">Go</a>
9h	I2CMDR	I2C Mode		<a href="#">Go</a>
Ah	I2CISRC	I2C Interrupt Source		<a href="#">Go</a>
Bh	I2CEMDR	I2C Extended Mode		<a href="#">Go</a>
Ch	I2CPSC	I2C Prescaler		<a href="#">Go</a>
20h	I2CFFTX	I2C FIFO Transmit		<a href="#">Go</a>
21h	I2CFFRX	I2C FIFO Receive		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 17-9 shows the codes that are used for access types in this section.

**Table 17-9. I2C\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		
i,j,k,l,m,n		When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula.
y		When this variable is used in a register name, an offset, or an address it refers to the value of a register array.

### 17.7.2.1 I2COAR Register (Offset = 0h) [Reset = 0000h]

I2COAR is shown in [Figure 17-20](#) and described in [Table 17-10](#).

Return to the [Summary Table](#).

The I2C own address register (I2COAR) is a 16-bit register. The I2C module uses this register to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 are used write 0s to bits 9-7.

**Figure 17-20. I2COAR Register**

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

**Table 17-10. 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. Reset type: SYSRSn

### 17.7.2.2 I2CIER Register (Offset = 1h) [Reset = 0000h]

I2CIER is shown in [Figure 17-21](#) and described in [Table 17-11](#).

Return to the [Summary Table](#).

I2CIER is used by the CPU to individually enable or disable I2C interrupt requests.

**Figure 17-21. 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						

**Table 17-11. I2CIER Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	AAS	R/W	0h	Addressed as slave interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
5	SCD	R/W	0h	Stop condition detected interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
4	XRDY	R/W	0h	Transmit-data-ready interrupt enable bit. This bit should not be set when using FIFO mode. Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
3	RRDY	R/W	0h	Receive-data-ready interrupt enable bit. This bit should not be set when using FIFO mode. Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
2	ARDY	R/W	0h	Register-access-ready interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
1	NACK	R/W	0h	No-acknowledgment interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled
0	ARBL	R/W	0h	Arbitration-lost interrupt enable Reset type: SYSRSn 0h (R/W) = Interrupt request disabled 1h (R/W) = Interrupt request enabled

### 17.7.2.3 I2CSTR Register (Offset = 2h) [Reset = 0410h]

I2CSTR is shown in [Figure 17-22](#) and described in [Table 17-12](#).

Return to the [Summary Table](#).

The I2C status register (I2CSTR) is a 16-bit register used to determine which interrupt has occurred and to read status information.

**Figure 17-22. I2CSTR Register**

15	14	13	12	11	10	9	8
RESERVED	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0h	R/W1C-0h	R/W1C-0h	R-0h	R-0h	R-1h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED	BYTESENT	SCD	XRDY	RRDY	ARDY	NACK	ARBL
R-0h	R/W1C-0h	R/W1C-0h	R-1h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

**Table 17-12. I2CSTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	SDIR	R/W1C	0h	Slave direction bit Reset type: SYSRSn 0h (R/W) = I2C is not addressed as a slave transmitter. SDIR is cleared by one of the following events: - It is manually cleared. To clear this bit, write a 1 to it. - Digital loopback mode is enabled. - A START or STOP condition occurs on the I2C bus. 1h (R/W) = I2C is addressed as a slave transmitter.
13	NACKSNT	R/W1C	0h	NACK sent bit. This bit is used when the I2C module is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used (see the description for NACKMOD in Reset type: SYSRSn 0h (R/W) = NACK not sent. NACKSNT bit is cleared by any one of the following events: - It is manually cleared. To clear this bit, write a 1 to it. - The I2C module is reset (either when 0 is written to the IRS bit of I2CMDR or when the whole device is reset). 1h (R/W) = NACK sent: A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB	R	0h	Bus busy bit. BB indicates whether the I2C-bus is busy or is free for another data transfer. See the paragraph following the table for more information Reset type: SYSRSn 0h (R/W) = Bus free. BB is cleared by any one of the following events: - The I2C module receives or transmits a STOP bit (bus free). - The I2C module is reset. 1h (R/W) = Bus busy: The I2C module has received or transmitted a START bit on the bus.

**Table 17-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	RSFULL	R	0h	<p>Receive shift register full bit.</p> <p>RSFULL indicates an overrun condition during reception. Overrun occurs when new data is received into the shift register (I2CRSR) and the old data has not been read from the receive register (I2CDRR). As new bits arrive from the SDA pin, they overwrite the bits in I2CRSR. The new data will not be copied to ICDRR until the previous data is read.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = No overrun detected. RSFULL is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = Overrun detected</p>
10	XSMT	R	1h	<p>Transmit shift register empty bit.</p> <p>XSMT = 0 indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (I2CXSR) is empty but the data transmit register (I2CDXR) has not been loaded since the last I2CDXR-to-I2CXSR transfer. The next I2CDXR-to-I2CXSR transfer will not occur until new data is in I2CDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Underflow detected (empty)</p> <p>1h (R/W) = No underflow detected (not empty). XSMT is set by one of the following events:</p> <ul style="list-style-type: none"> <li>- Data is written to I2CDXR.</li> <li>- The I2C module is reset</li> </ul>
9	AAS	R	0h	<p>Addressed-as-slave bit</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the 7-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or a repeated START condition. In the 10-bit addressing mode, the AAS bit is cleared when receiving a NACK, a STOP condition, or by a slave address different from the I2C peripheral's own slave address.</p> <p>1h (R/W) = The I2C module has recognized its own slave address or an address of all zeros (general call).</p>
8	AD0	R	0h	<p>Address 0 bits</p> <p>Reset type: SYSRSn</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	RESERVED	R	0h	Reserved
6	BYTESENT	R/W1C	0h	<p>Byte Transmit over indication.</p> <p>BYTESENT is set when the master/slave has successfully sent the byte on SCL/SDA lines. This is diagnostic register which needs to be explicitly cleared by Software. In case not cleared the stale status would keep reflecting as no automated clear incorporated to avoid corner conditions.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module has not finished transmitting the next data byte. BYTESENT is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- It is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = The I2C module has completed the transmission of a byte.</p>

**Table 17-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	SCD	R/W1C	0h	<p>Stop condition detected bit.</p> <p>SCD is set when the I2C sends or receives a STOP condition. The I2C module delays clearing of the I2CMDR[STP] bit until the SCD bit is set.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = STOP condition not detected since SCD was last cleared. SCD is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CISRC is read by the CPU when it contains the value 110b (stop condition detected). Emulator reads of the I2CISRC do not affect this bit.</li> <li>- SCD is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = A STOP condition has been detected on the I2C bus.</p>
4	XRDY	R	1h	<p>Transmit-data-ready interrupt flag bit. When not in FIFO mode, XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data.</p> <p>FCM=0 : When the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request When in FIFO mode, use TXFFINT instead.</p> <p>FCM=1: XRDY is asserted only when next data is required it gets deasserted with write to I2CDXR. Both Polling and interrupt based data transfers are allowed in the FCM mode.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = I2CDXR not ready. XRDY is cleared when data is written to I2CDXR.</p> <p>1h (R/W) = I2CDXR ready: Data has been copied from I2CDXR to I2CXSR.</p> <p>XRDY is also forced to 1 when the I2C module is reset.</p>
3	RRDY	R/W1C	0h	<p>Receive-data-ready interrupt flag bit.</p> <p>When not in FIFO mode, RRDY indicates that the data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR. The CPU can poll RRDY or use the RRDY interrupt request When in FIFO mode, use RXFFINT instead.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = I2CDRR not ready. RRDY is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- I2CDRR is read by the CPU. Emulator reads of the I2CDRR do not affect this bit.</li> <li>- RRDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = I2CDRR ready: Data has been copied from I2CRSR to I2CDRR.</p>
2	ARDY	R/W1C	0h	<p>Register-access-ready interrupt flag bit (only Applicable when the I2C module is in the master mode).</p> <p>ARDY indicates that the I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The registers are not ready to be accessed. ARDY is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- The I2C module starts using the current register contents.</li> <li>- ARDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = The registers are ready to be accessed.</p> <p>In the nonrepeat mode (RM = 0 in I2CMDR): If STP = 0 in I2CMDR, the ARDY bit is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C module generates a STOP condition when the counter reaches 0).</p> <p>In the repeat mode (RM = 1): ARDY is set at the end of each byte transmitted from I2CDXR.</p>

**Table 17-12. I2CSTR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	NACK	R/W1C	0h	<p>No-acknowledgment interrupt flag bit.</p> <p>NACK applies when the I2C module is a master transmitter. NACK indicates whether the I2C module has detected an acknowledge bit (ACK) or a noacknowledge bit (NACK) from the slave receiver. The CPU can poll NACK or use the NACK interrupt request.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = ACK received/NACK not received. This bit is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- An acknowledge bit (ACK) has been sent by the slave receiver.</li> <li>- NACK is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The CPU reads the interrupt source register (I2CISRC) and the register contains the code for a NACK interrupt. Emulator reads of the I2CISRC do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = NACK bit received. The hardware detects that a no-acknowledge (NACK) bit has been received.</p> <p>Note: While the I2C module performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.</p>
0	ARBL	R/W1C	0h	<p>Arbitration-lost interrupt flag bit (only applicable when the I2C module is a master-transmitter).</p> <p>ARBL primarily indicates when the I2C module has lost an arbitration contest with another master/transmitter. The CPU can poll ARBL or use the ARBL interrupt request.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Arbitration not lost. AL is cleared by any one of the following events:</p> <ul style="list-style-type: none"> <li>- AL is manually cleared. To clear this bit, write a 1 to it.</li> <li>- The CPU reads the interrupt source register (I2CISRC) and the register contains the code for an AL interrupt. Emulator reads of the I2CISRC do not affect this bit.</li> <li>- The I2C module is reset.</li> </ul> <p>1h (R/W) = Arbitration lost. AL is set by any one of the following events:</p> <ul style="list-style-type: none"> <li>- The I2C module senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously.</li> <li>- The I2C module attempts to start a transfer while the BB (bus busy) bit is set to 1.</li> </ul> <p>When AL becomes 1, the MST and STP bits of I2CMMDR are cleared, and the I2C module becomes a slave-receiver.</p>

### 17.7.2.4 I2CCLKL Register (Offset = 3h) [Reset = 0000h]

I2CCLKL is shown in [Figure 17-23](#) and described in [Table 17-13](#).

Return to the [Summary Table](#).

I2C Clock low-time divider

**Figure 17-23. I2CCLKL Register**

15	14	13	12	11	10	9	8
I2CCLKL							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCLKL							
R/W-0h							

**Table 17-13. I2CCLKL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCLKL	R/W	0h	<p>Clock low-time divide-down value.</p> <p>To produce the low time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is an adjustment factor based on the prescaler. See the Clock Divider Registers section of the Introduction for details.</p> <p>Note: These bits must be set to a non-zero value for proper I2C clock generation.</p> <p>Reset type: SYSRSn</p>

### 17.7.2.5 I2CCLKH Register (Offset = 4h) [Reset = 0000h]

I2CCLKH is shown in [Figure 17-24](#) and described in [Table 17-14](#).

Return to the [Summary Table](#).

I2C Clock high-time divider

**Figure 17-24. I2CCLKH Register**

15	14	13	12	11	10	9	8
I2CCLKH							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCLKH							
R/W-0h							

**Table 17-14. I2CCLKH Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCLKH	R/W	0h	<p>Clock high-time divide-down value.</p> <p>To produce the high time duration of the master clock, the period of the module clock is multiplied by (ICCL + d). d is an adjustment factor based on the prescaler. See the Clock Divider Registers section of the Introduction for details.</p> <p>Note: These bits must be set to a non-zero value for proper I2C clock generation.</p> <p>Reset type: SYSRSn</p>

### 17.7.2.6 I2CCNT Register (Offset = 5h) [Reset = 0000h]

I2CCNT is shown in [Figure 17-25](#) and described in [Table 17-15](#).

Return to the [Summary Table](#).

I2CCNT is a 16-bit register used to indicate how many data bytes to transfer when the I2C module is configured as a transmitter, or to receive when configured as a master receiver. In the repeat mode (RM = 1), I2CCNT is not used.

The value written to I2CCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each byte transferred (I2CCNT remains unchanged). If a STOP condition is requested in the master mode (STP = 1 in I2CMDR), the I2C module terminates the transfer with a STOP condition when the countdown is complete (that is, when the last byte has been transferred).

**Figure 17-25. I2CCNT Register**

15	14	13	12	11	10	9	8
I2CCNT							
R/W-0h							
7	6	5	4	3	2	1	0
I2CCNT							
R/W-0h							

**Table 17-15. I2CCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	I2CCNT	R/W	0h	<p>Data count value. I2CCNT indicates the number of data bytes to transfer or receive.</p> <p>If a STOP condition is specified (STP=1) then I2CCNT will decrease after each byte is sent until it reaches zero, which in turn will generate a STOP condition.</p> <p>The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = data count value is 65536</p> <p>1h (R/W) = data count value is 1</p> <p>2h (R/W) = data count value is 2</p> <p>FFFFh (R/W) = data count value is 65535</p>

### 17.7.2.7 I2CDRR Register (Offset = 6h) [Reset = 0000h]

I2CDRR is shown in [Figure 17-26](#) and described in [Table 17-16](#).

Return to the [Summary Table](#).

I2CDRR is a 16-bit register used by the CPU to read received data. The I2C module can receive a data byte with 1 to 8 bits. The number of bits is selected with the bit count (BC) bits in I2CMADR. One bit at a time is shifted in from the SDA pin to the receive shift register (I2CRSR). When a complete data byte has been received, the I2C module copies the data byte from I2CRSR to I2CDRR. The CPU cannot access I2CRSR directly.

If a data byte with fewer than 8 bits is in I2CDRR, the data value is right-justified, and the other bits of I2CDRR(7-0) are undefined. For example, if BC = 011 (3-bit data size), the receive data is in I2CDRR(2-0), and the content of I2CDRR(7-3) is undefined.

When in the receive FIFO mode, the I2CDRR register acts as the receive FIFO buffer.

**Figure 17-26. I2CDRR Register**

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

**Table 17-16. I2CDRR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R	0h	Receive data Reset type: SYSRSn

### 17.7.2.8 I2CSAR Register (Offset = 7h) [Reset = 03FFh]

I2CSAR is shown in [Figure 17-27](#) and described in [Table 17-17](#).

Return to the [Summary Table](#).

The I2C slave address register (I2CSAR) is a 16-bit register for storing the next slave address that will be transmitted by the I2C module when it is a master. The SAR field of I2CSAR contains a 7-bit or 10-bit slave address. When the I2C module is not using the free data format (FDF = 0 in I2CMDR), it uses this address to initiate data transfers with a slave, or slaves. When the address is nonzero, the address is for a particular slave. When the address is 0, the address is a general call to all slaves. If the 7-bit addressing mode is selected (XA = 0 in I2CMDR), only bits 6-0 of I2CSAR are used write 0s to bits 9-7.

**Figure 17-27. I2CSAR Register**

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

**Table 17-17. I2CSAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	SAR	R/W	3FFh	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. In 10-bit addressing mode (XA = 1 in I2CMDR): 000h-3FFh Bits 9-0 provide the 10-bit slave address that the I2C module transmits when it is in the master transmitter mode. Reset type: SYSRSn

### 17.7.2.9 I2CDXR Register (Offset = 8h) [Reset = 0000h]

I2CDXR is shown in [Figure 17-28](#) and described in [Table 17-18](#).

Return to the [Summary Table](#).

The CPU writes transmit data to I2CDXR. This 16-bit register accepts a data byte with 1 to 8 bits. Before writing to I2CDXR, specify how many bits are in a data byte by loading the appropriate value into the bit count (BC) bits of I2CMR. When writing a data byte with fewer than 8 bits, make sure the value is right-aligned in I2CDXR. After a data byte is written to I2CDXR, the I2C module copies the data byte to the transmit shift register (I2CXSR). The CPU cannot access I2CXSR directly. From I2CXSR, the I2C module shifts the data byte out on the SDA pin, one bit at a time.

When in the transmit FIFO mode, the I2CDXR register acts as the transmit FIFO buffer.

**Figure 17-28. I2CDXR Register**

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

**Table 17-18. I2CDXR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	DATA	R/W	0h	Transmit data Reset type: SYSRSn

### 17.7.2.10 I2CMR Register (Offset = 9h) [Reset = 0000h]

I2CMR is shown in [Figure 17-29](#) and described in [Table 17-19](#).

Return to the [Summary Table](#).

The I2C mode register (I2CMR) is a 16-bit register that contains the control bits of the I2C module.

**Figure 17-29. I2CMR Register**

15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	RESERVED	STP	MST	TRX	XA
R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RM	DLB	IRS	STB	FDL		BC	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h	

**Table 17-19. I2CMR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	NACKMOD	R/W	0h	<p>NACK mode bit. This bit is only applicable when the I2C module is acting as a receiver.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the slave-receiver mode: The I2C module sends an acknowledge (ACK) bit to the transmitter during each acknowledge cycle on the bus. The I2C module only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit.</p> <p>In the master-receiver mode: The I2C module sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. At that point, the I2C module sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit</p> <p>1h (R/W) = In either slave-receiver or master-receiver mode: The I2C module sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared.</p> <p>Important: To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.</p>
14	FREE	R/W	0h	<p>This bit controls the action taken by the I2C module when a debugger breakpoint is encountered.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = When I2C module is master: If SCL is low when the breakpoint occurs, the I2C module stops immediately and keeps driving SCL low, whether the I2C module is the transmitter or the receiver. If SCL is high, the I2C module waits until SCL becomes low and then stops.</p> <p>When I2C module is slave: A breakpoint forces the I2C module to stop when the current transmission/reception is complete.</p> <p>1h (R/W) = The I2C module runs free that is, it continues to operate when a breakpoint occurs.</p>
13	STT	R/W	0h	<p>START condition bit (only applicable when the I2C module is a master). The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions (see Table 9-6). Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS = 0.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = In the master mode, STT is automatically cleared after the START condition has been generated.</p> <p>1h (R/W) = In the master mode, setting STT to 1 causes the I2C module to generate a START condition on the I2C-bus</p>
12	RESERVED	R	0h	Reserved

**Table 17-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	STP	R/W	0h	<p>STOP condition bit (only applicable when the I2C module is a master).</p> <p>In the master mode, the RM,STT, and STP bits determine when the I2C module starts and stops data transmissions.</p> <p>Note that the STT and STP bits can be used to terminate the repeat mode, and that this bit is not writable when IRS=0. When in non-repeat mode, at least one byte must be transferred before a stop condition can be generated. The I2C module delays clearing of this bit until after the I2CSTR[SCD] bit is set. To avoid disrupting the I2C state machine, the user must wait until this bit is clear before initiating a new message.</p> <p>Reset type: SYSRSn</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>Reset type: SYSRSn</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>Reset type: SYSRSn</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>Reset type: SYSRSn</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 or master-receiver).</p> <p>The RM, STT, and STP bits determine when the I2C module starts and stops data transmissions</p> <p>Reset type: SYSRSn</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 17-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	DLB	R/W	0h	<p>Digital loopback mode bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Digital loopback mode is disabled.</p> <p>1h (R/W) = Digital loopback mode is enabled. For proper operation in this mode, the MST bit must be 1.</p> <p>In the digital loopback mode, data transmitted out of I2CDXR is received in I2CDRR after n device cycles by an internal path, where: <math>n = ((I2C \text{ input clock frequency/module clock frequency}) \times 8)</math></p> <p>The transmit clock is also the receive clock. The address transmitted on the SDA pin is the address in I2COAR.</p> <p>Note: The free data format (FDF = 1) is not supported in the digital loopback mode.</p>
5	IRS	R/W	0h	<p>I2C module reset bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module is in reset/disabled. When this bit is cleared to 0, all status bits (in I2CSTR) are set to their default values.</p> <p>1h (R/W) = The I2C module is enabled. This has the effect of releasing the I2C bus if the I2C peripheral is holding it.</p>
4	STB	R/W	0h	<p>START byte mode bit. This bit is only applicable when the I2C module is a master. As described in version 2.1 of the Philips Semiconductors I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C module is a slave, it ignores a START byte from a master, regardless of the value of the STB bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The I2C module is not in the START byte mode.</p> <p>1h (R/W) = The I2C module is in the START byte mode. When you set the START condition bit (STT), the I2C module begins the transfer with more than just a START condition. Specifically, it generates:</p> <ol style="list-style-type: none"> <li>1. A START condition</li> <li>2. A START byte (0000 0001b)</li> <li>3. A dummy acknowledge clock pulse</li> <li>4. A repeated START condition</li> </ol> <p>Then, as normal, the I2C module sends the slave address that is in I2CSAR.</p>
3	FDF	R/W	0h	<p>Free data format mode bit.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.</p> <p>1h (R/W) = Free data format mode is enabled. Transfers have the free data (no address) format described in Section 9.2.5.</p> <p>The free data format is not supported in the digital loopback mode (DLB=1).</p>

**Table 17-19. I2CMDR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2-0	BC	R/W	0h	Bit count bits. 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. 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 Reset type: SYSRSn 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

### 17.7.2.11 I2CISRC Register (Offset = Ah) [Reset = 0000h]

I2CISRC is shown in [Figure 17-30](#) and described in [Table 17-20](#).

Return to the [Summary Table](#).

The I2C interrupt source register (I2CISRC) is a 16-bit register used by the CPU to determine which event generated the I2C interrupt.

**Figure 17-30. I2CISRC Register**

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

**Table 17-20. I2CISRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-8	WRITE_ZEROS	R/W	0h	TI internal testing bits These reserved bit locations should always be written as zeros. Reset type: SYSRSn
7-3	RESERVED	R	0h	Reserved
2-0	INTCODE	R	0h	Interrupt code bits. The binary code in INTCODE indicates the event that generated an I2C interrupt. A CPU read will clear this field. If another lower priority interrupt is pending and enabled, the value corresponding to that interrupt will then be loaded. Otherwise, the value will stay cleared. The interrupt events below are listed in descending order of priority. That is INTCODE 1 (Arbitration lost) has the highest priority and INTCODE 7 (Addressed as slave) has the lowest priority. In the case of an arbitration lost, a no-acknowledgment condition detected, or a stop condition detected, a CPU read will also clear the associated interrupt flag bit in the I2CSTR register. Emulator reads will not affect the state of this field or of the status bits in the I2CSTR register. Reset type: SYSRSn 0h (R/W) = None 1h (R/W) = Arbitration lost 2h (R/W) = No-acknowledgment condition detected 3h (R/W) = Registers ready to be accessed 4h (R/W) = Receive data ready 5h (R/W) = Transmit data ready 6h (R/W) = Stop condition detected 7h (R/W) = Addressed as slave

**17.7.2.12 I2CEMDR Register (Offset = Bh) [Reset = 0001h]**

I2CEMDR is shown in [Figure 17-31](#) and described in [Table 17-21](#).

Return to the [Summary Table](#).

I2C Extended Mode

**Figure 17-31. I2CEMDR Register**

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

**Table 17-21. I2CEMDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	FCM	R/W	0h	<p>Forward Compatibility mode.</p> <p>This bit when programmed brings the functionality of Tx request only when Tx data required regardless of data status in Tx buffer for non-FIFO mode.</p> <p>This register affects the XRDY behavior hence needs to be set after releasing the IRS (I2CMDR[5]).</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Legacy functionality of requesting Tx data upon buffer copy to shift register or upon start condition is active. Stale data is reused after illegal start, ARB Lost, NACK conditions.</p> <p>1h (R/W) = New functionality of requesting data only upon ACK (address/data) is active.</p>
0	BC	R/W	1h	<p>Backwards compatibility mode.</p> <p>This bit affects the timing of the transmit status bits (XRDY and XSMT) in the I2CSTR register when in slave transmitter mode.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = See the 'Backwards Compatibility Mode Bit, Slave Transmitter' Figure for details.</p> <p>1h (R/W) = See the 'Backwards Compatibility Mode Bit, Slave Transmitter' Figure for details.</p>

### 17.7.2.13 I2CPSC Register (Offset = Ch) [Reset = 0000h]

I2CPSC is shown in [Figure 17-32](#) and described in [Table 17-22](#).

Return to the [Summary Table](#).

The I2C prescaler register (I2CPSC) is a 16-bit register (see [Figure 14-21](#)) used for dividing down the I2C input clock to obtain the desired module clock for the operation of the I2C module. See the device-specific data manual for the supported range of values for the module clock frequency.

IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

**Figure 17-32. I2CPSC Register**

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

**Table 17-22. I2CPSC Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	IPSC	R/W	0h	I2C prescaler divide-down value. IPSC determines how much the CPU clock is divided to create the module clock of the I2C module: module clock frequency = I2C input clock frequency / (IPSC + 1) Note: IPSC must be initialized while the I2C module is in reset (IRS = 0 in I2CMDR). Reset type: SYSRSn

### 17.7.2.14 I2CFFTX Register (Offset = 20h) [Reset = 0000h]

I2CFFTX is shown in [Figure 17-33](#) and described in [Table 17-23](#).

Return to the [Summary Table](#).

The I2C transmit FIFO register (I2CFFTX) is a 16-bit register that contains the I2C FIFO mode enable bit as well as the control and status bits for the transmit FIFO mode of operation on the I2C peripheral.

**Figure 17-33. I2CFFTX Register**

15	14	13	12	11	10	9	8	
RESERVED	I2CFFEN	TXFFRST	TXFFST					
R-0h	R/W-0h	R/W-0h	R-0h					
7	6	5	4	3	2	1	0	
TXFFINT	TXFFINTCLR	TXFFIENA	TXFFIL					
R-0h	R-0/W1S-0h	R/W-0h	R/W-0h					

**Table 17-23. I2CFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RESERVED	R	0h	Reserved
14	I2CFFEN	R/W	0h	I2C FIFO mode enable bit. This bit must be enabled for either the transmit or the receive FIFO to operate correctly. Reset type: SYSRSn 0h (R/W) = Disable the I2C FIFO mode. 1h (R/W) = Enable the I2C FIFO mode.
13	TXFFRST	R/W	0h	Transmit FIFO Reset Reset type: SYSRSn 0h (R/W) = Reset the transmit FIFO pointer to 0000 and hold the transmit FIFO in the reset state. 1h (R/W) = Enable the transmit FIFO operation.
12-8	TXFFST	R	0h	Contains the status of the transmit FIFO: xxxxx Transmit FIFO contains xxxxx bytes. 00000 Transmit FIFO is empty. Note: Since these bits are reset to zero, the transmit FIFO interrupt flag will be set when the transmit FIFO operation is enabled and the I2C is taken out of reset. This will generate a transmit FIFO interrupt if enabled. To avoid any detrimental effects from this, write a one to the TXFFINTCLR once the transmit FIFO operation is enabled and the I2C is taken out of reset. Reset type: SYSRSn
7	TXFFINT	R	0h	Transmit FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the TXFFINTCLR bit. If the TXFFIENA bit is set, this bit will generate an interrupt when it is set. Reset type: SYSRSn 0h (R/W) = Transmit FIFO interrupt condition has not occurred. 1h (R/W) = Transmit FIFO interrupt condition has occurred.
6	TXFFINTCLR	R-0/W1S	0h	Transmit FIFO Interrupt Flag Clear Reset type: SYSRSn 0h (R/W) = Writes of zeros have no effect. Reads return a 0. 1h (R/W) = Writing a 1 to this bit clears the TXFFINT flag.
5	TXFFIENA	R/W	0h	Transmit FIFO Interrupt Enable Reset type: SYSRSn 0h (R/W) = Disabled. TXFFINT flag does not generate an interrupt when set. 1h (R/W) = Enabled. TXFFINT flag does generate an interrupt when set.

**Table 17-23. I2CFFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 this device has a 16-level transmit FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels. Reset type: SYSRSn

### 17.7.2.15 I2CFFRX Register (Offset = 21h) [Reset = 0000h]

I2CFFRX is shown in [Figure 17-34](#) and described in [Table 17-24](#).

Return to the [Summary Table](#).

The I2C receive FIFO register (I2CFFRX) is a 16-bit register that contains the control and status bits for the receive FIFO mode of operation on the I2C peripheral.

**Figure 17-34. I2CFFRX Register**

15	14	13	12	11	10	9	8	
RESERVED		RXFFRST	RXFFST					
R-0h		R/W-0h	R-0h					
7	6	5	4	3	2	1	0	
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL					
R-0h	R-0/W1S-0h	R/W-0h	R/W-0h					

**Table 17-24. I2CFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-14	RESERVED	R	0h	Reserved
13	RXFFRST	R/W	0h	I2C receive FIFO reset bit Reset type: SYSRSn 0h (R/W) = Reset the receive FIFO pointer to 0000 and hold the receive FIFO in the reset state. 1h (R/W) = Enable the receive FIFO operation.
12-8	RXFFST	R	0h	Contains the status of the receive FIFO: xxxxx Receive FIFO contains xxxxx bytes 00000 Receive FIFO is empty. Reset type: SYSRSn
7	RXFFINT	R	0h	Receive FIFO interrupt flag. This bit cleared by a CPU write of a 1 to the RXFFINTCLR bit. If the RXFFIENA bit is set, this bit will generate an interrupt when it is set Reset type: SYSRSn 0h (R/W) = Receive FIFO interrupt condition has not occurred. 1h (R/W) = Receive FIFO interrupt condition has occurred.
6	RXFFINTCLR	R-0/W1S	0h	Receive FIFO interrupt flag clear bit. Reset type: SYSRSn 0h (R/W) = Writes of zeros have no effect. Reads return a zero. 1h (R/W) = Writing a 1 to this bit clears the RXFFINT flag.
5	RXFFIENA	R/W	0h	Receive FIFO interrupt enable bit. Reset type: SYSRSn 0h (R/W) = Disabled. RXFFINT flag does not generate an interrupt when set. 1h (R/W) = Enabled. RXFFINT flag does generate an interrupt when set.
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 this device has a 16-level receive FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels. Reset type: SYSRSn

### 17.7.3 I2C Registers to Driverlib Functions

**Table 17-25. I2C Registers to Driverlib Functions**

File	Driverlib Function
<b>OAR</b>	
i2c.h	I2C_setOwnAddress
<b>IER</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
<b>STR</b>	
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_isBusBusy
i2c.h	I2C_getStatus
i2c.h	I2C_clearStatus
<b>CLKL</b>	
i2c.c	I2C_initController
i2c.c	I2C_initControllerModuleFrequency
<b>CLKH</b>	
i2c.c	I2C_initController
i2c.c	I2C_initControllerModuleFrequency
<b>CNT</b>	
i2c.h	I2C_setDataCount
<b>DRR</b>	
i2c.h	I2C_getData
<b>TAR</b>	
i2c.h	I2C_setTargetAddress
<b>DXR</b>	
i2c.h	I2C_putData
<b>MDR</b>	
i2c.h	I2C_enableModule
i2c.h	I2C_disableModule
i2c.h	I2C_setConfig
i2c.h	I2C_setBitCount
i2c.h	I2C_sendStartCondition
i2c.h	I2C_sendStopCondition
i2c.h	I2C_sendNACK
i2c.h	I2C_getStopConditionStatus
i2c.h	I2C_setAddressMode
i2c.h	I2C_setEmulationMode
i2c.h	I2C_enableLoopback
i2c.h	I2C_disableLoopback
<b>ISRC</b>	
i2c.h	I2C_getInterruptSource
<b>EMDR</b>	
i2c.h	I2C_setExtendedMode
<b>PSC</b>	

**Table 17-25. I2C Registers to Driverlib Functions (continued)**

File	Driverlib Function
i2c.c	I2C_initController
i2c.c	I2C_initControllerModuleFrequency
i2c.c	I2C_configureModuleFrequency
i2c.c	I2C_configureModuleClockFrequency
i2c.h	I2C_getPreScaler
<b>FFTX</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_enableFIFO
i2c.h	I2C_disableFIFO
i2c.h	I2C_setFIFOInterruptLevel
i2c.h	I2C_getFIFOInterruptLevel
i2c.h	I2C_getTxFIFOStatus
<b>FFRX</b>	
i2c.c	I2C_enableInterrupt
i2c.c	I2C_disableInterrupt
i2c.c	I2C_getInterruptStatus
i2c.c	I2C_clearInterruptStatus
i2c.h	I2C_enableFIFO
i2c.h	I2C_disableFIFO
i2c.h	I2C_setFIFOInterruptLevel
i2c.h	I2C_getFIFOInterruptLevel
i2c.h	I2C_getRxFIFOStatus

## Chapter 18

# Serial Communications Interface (SCI)



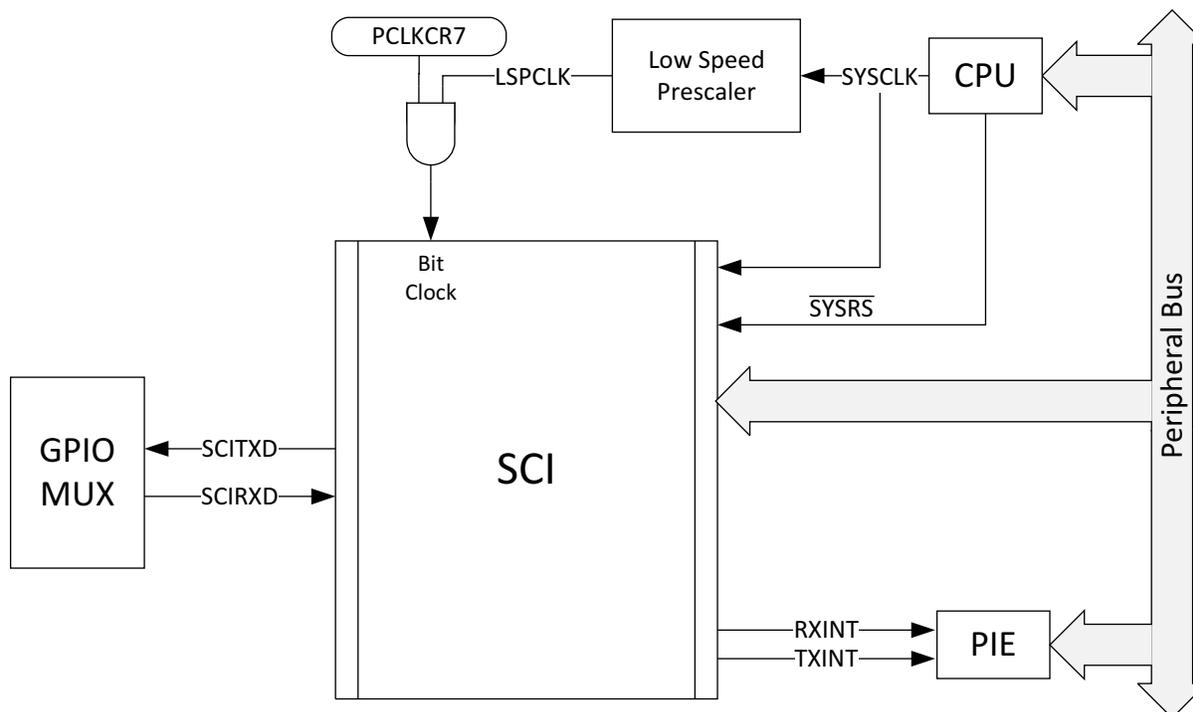
This chapter describes the features and operation of the serial communication interface (SCI) module. SCI is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 16-level deep FIFO for reducing servicing overhead, and each has a 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.

<b>18.1 Introduction</b> .....	1784
<b>18.2 Architecture</b> .....	1785
<b>18.3 SCI Module Signal Summary</b> .....	1785
<b>18.4 Configuring Device Pins</b> .....	1787
<b>18.5 Multiprocessor and Asynchronous Communication Modes</b> .....	1787
<b>18.6 SCI Programmable Data Format</b> .....	1788
<b>18.7 SCI Multiprocessor Communication</b> .....	1789
<b>18.8 Idle-Line Multiprocessor Mode</b> .....	1790
<b>18.9 Address-Bit Multiprocessor Mode</b> .....	1792
<b>18.10 SCI Communication Format</b> .....	1793
<b>18.11 SCI Port Interrupts</b> .....	1796
<b>18.12 SCI Baud Rate Calculations</b> .....	1798
<b>18.13 SCI Enhanced Features</b> .....	1799
<b>18.14 Software</b> .....	1802
<b>18.15 SCI Registers</b> .....	1804

## 18.1 Introduction

The SCI interfaces are shown in [Figure 18-1](#).



**Figure 18-1. SCI CPU Interface**

### 18.1.1 Features

Features of the SCI module include:

- Two external pins (both pins can be used as GPIO, if not used for SCI):
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin
- Baud rate programmable to 64K different rates
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits
  - An extra bit to distinguish addresses from data (address bit mode only)
- Four error-detection flags: parity, overrun, framing, and break detection
- Two wake-up multiprocessor modes: idle-line and address bit
- Half- or full-duplex operation
- Double-buffered receive and transmit functions
- Transmitter and receiver operations can be accomplished through interrupt-driven or polled algorithms with status flags
- Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- NRZ (non-return-to-zero) format

Enhanced features include:

- Auto-baud-detect hardware logic
- 16-level transmit/receive FIFO

## 18.1.2 SCI Related Collateral

### Foundational Materials

- [C2000 Academy - SCI](#)
- [One Minute RS-485 Introduction \(Video\)](#)
- [RS-232, RS-422, RS-485: What Are the Differences? \(Video\)](#)

### Getting Started Materials

- [\[FAQ\] My C2000 SCI is not Transmitting and/or Receiving data correctly, how do I fix this?](#)

## 18.1.3 Block Diagram

Figure 18-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Section 18.15.

## 18.2 Architecture

The major elements used in full-duplex operation are shown in Figure 18-2 and include:

- A transmitter (TX) and the 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 the 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
- Control and status registers

The SCI receiver and transmitter can operate either independently or simultaneously.

## 18.3 SCI Module Signal Summary

A summarized description of each SCI signal name is shown in Table 18-1.

**Table 18-1. SCI Module Signal Summary**

Signal Name	Description
<b>External signals</b>	
SCIRXD	SCI Asynchronous Serial Port receive data
SCITXD	SCI Asynchronous Serial Port transmit data
<b>Control</b>	
Baud clock	LSPCLK Prescaled clock
<b>Interrupt signals</b>	
TXINT	Transmit interrupt
RXINT	Receive interrupt

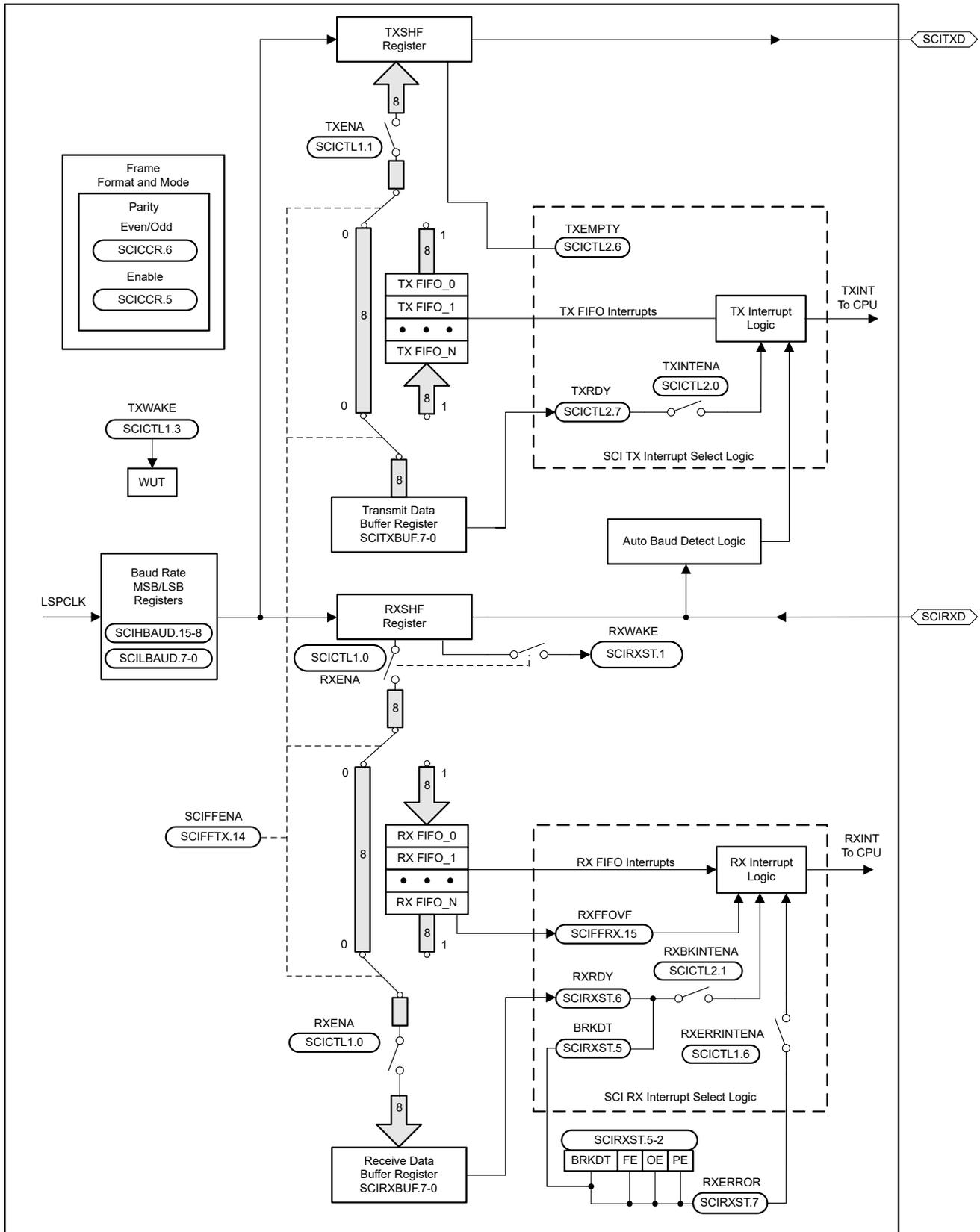


Figure 18-2. Serial Communications Interface (SCI) Module Block Diagram

## 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 must 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 *General-Purpose Input/Output (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:

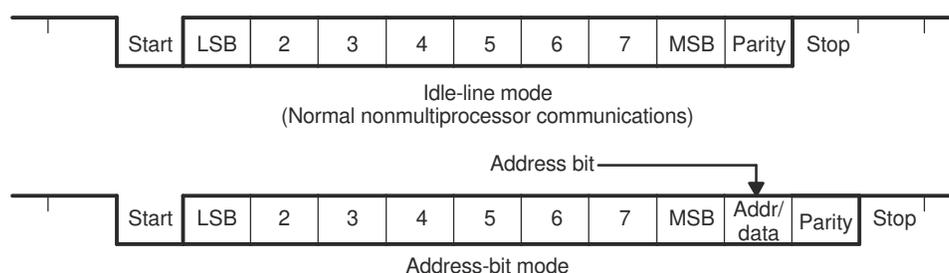
- One start bit
- One to eight data bits
- An even/odd parity bit or no parity bit
- One or two stop bits

## 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 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**

Bits	Bit Name	Designation	Functions
2-0	SCICCHAR	SCICCR.2:0	Select the character (data) length (one to eight bits).
5	PARITYENA (ENABLE)	SCICCR.5	Enables the parity function if set to 1, or disables the parity function if cleared to 0.
6	PARITY (EVEN/ODD)	SCICCR.6	If parity is enabled, selects odd parity if cleared to 0; even parity if set to 1.
7	STOPBITS	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 can 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 the processor is interrupted only when the address byte is detected. When the 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, the receiver 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 10 bytes of data. The idle-line mode must 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, this mode does not 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 using 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.

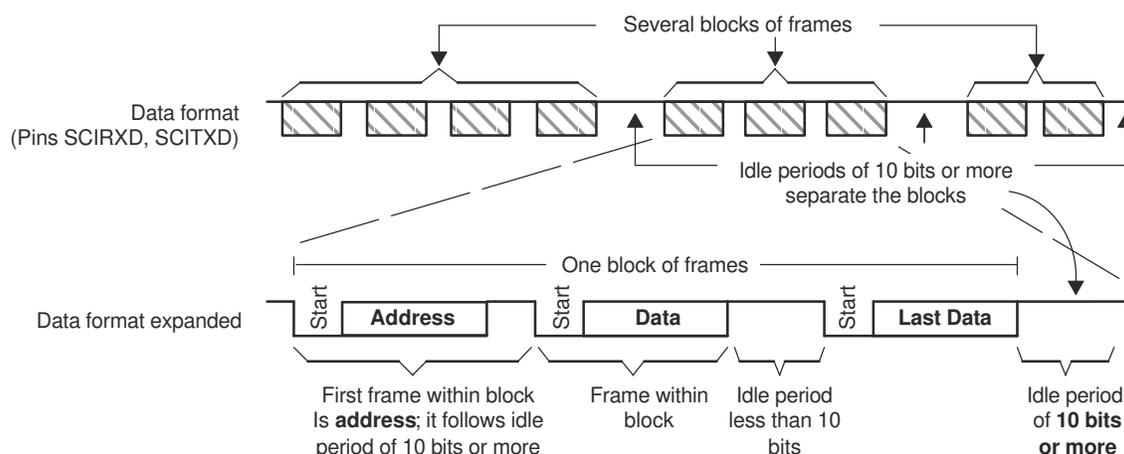
#### 18.7.3 Receipt Sequence

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 in non-FIFO mode of operation. In FIFO mode, RXFFINT serves this purpose and to enable this, RXFFINTEN in SCIFFRX register must be enabled with RXFFIL in the same register set to 1). The SCI 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 the 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:

1. SCI wakes up after receipt of the block-start signal.
2. The processor recognizes the next SCI interrupt.
3. The interrupt service routine compares the received address (sent by a remote transmitter) to the ISR address.
4. If the CPU is being addressed, the service routine clears the SLEEP bit and receives the rest of the data block.
5. If the CPU is not being addressed, the SLEEP bit remains set. This lets the CPU continue to execute the main program without being interrupted by the SCI port until the next detection of a block start.

#### Note

In IDLE mode, if the SCI is taking greater than 10 bit periods to read all the RXDATA from the FIFO, the SCI can miss the immediate block start to be detected.

The RXWAKE logic asserts only one time when the SCI identifies 10 bit periods of IDLE. The SCI does not assert again if RXBUF is read (which clears the WAKE condition) even if the line continues to be idle after RXBUF read.

So, if the ISR is taking more than 10 bit periods of time to read all the RXDATA from the FIFO using RXBUF, the SCI can miss to detect the next block start. This is applicable for both FIFO and Non-FIFO mode when the CPU takes greater than 10 bit clocks of SCI to read the data from RXBUF/ FIFO.

To avoid this, either of the following is recommended:

- Set SCICTL1.SWRESET after reading all RX data at the end of the ISR.
- Read and check RXWAKE status bit before reading the RXBUF register. If RXWAKE is set, don't set SLEEP bit for RX at the end of the ISR.

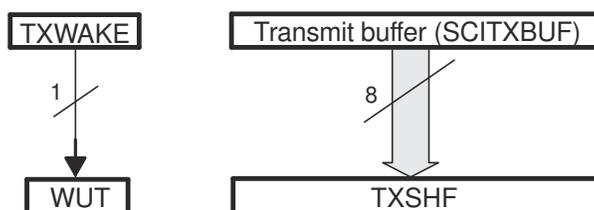
## 18.8.2 Block Start Signal

There are two ways to send a block-start signal:

- **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.
- **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**

### 18.8.3.1 Sending a Block Start Signal

To send out a block-start signal of exactly one frame time during a sequence of block transmissions:

1. Write a 1 to the TXWAKE bit.
2. Write a data word (content not important: a don't care) to the SCITXBUF register (transmit data buffer) to send a block-start signal. (The first data word written is suppressed while the block-start signal is sent out and ignored after that.) When the TXSHF (transmit shift register) is free again, SCITXBUF contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared.

Because TXWAKE was set to a 1, the start, data, and parity bits are replaced by an idle period of 11 bits transmitted following the last stop bit of the previous frame.

3. Write a new address value to SCITXBUF.

A don't-care data word must first be written to register SCITXBUF so that the TXWAKE bit value can be shifted to WUT. After the don't-care data word is shifted to the TXSHF register, the SCITXBUF (and TXWAKE, if necessary) can be written to again because TXSHF and WUT are both double-buffered.

## 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:

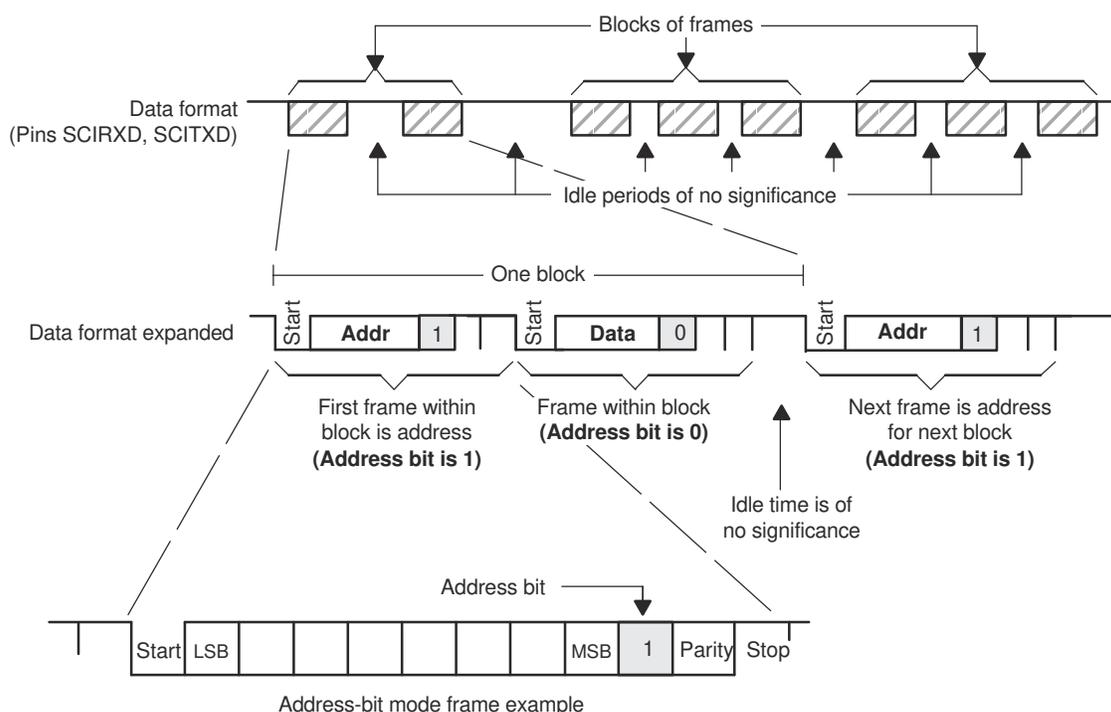
1. Set the TXWAKE bit to 1 and write the appropriate address value to the SCITXBUF register.

When this address value is transferred to the TXSHF register and shifted out, the 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 to frames, the external transmitting and receiving devices do not use a synchronized serial clock. The clock can be generated locally.

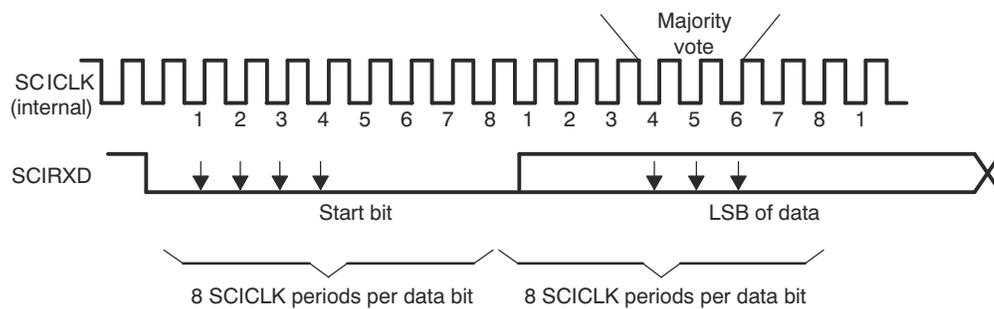
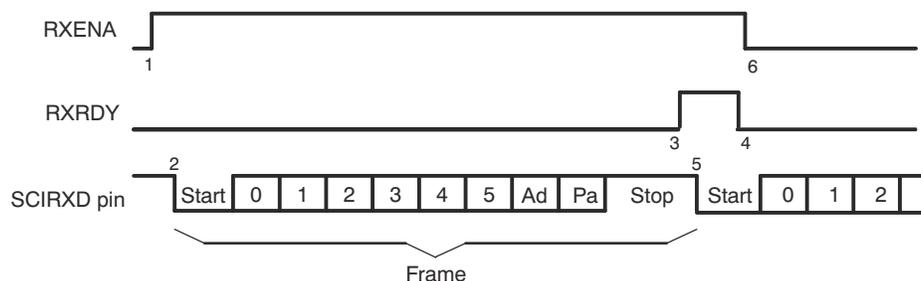


Figure 18-7. SCI Asynchronous Communications Format

### 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



**Figure 18-8. SCI RX Signals in Communication Modes**

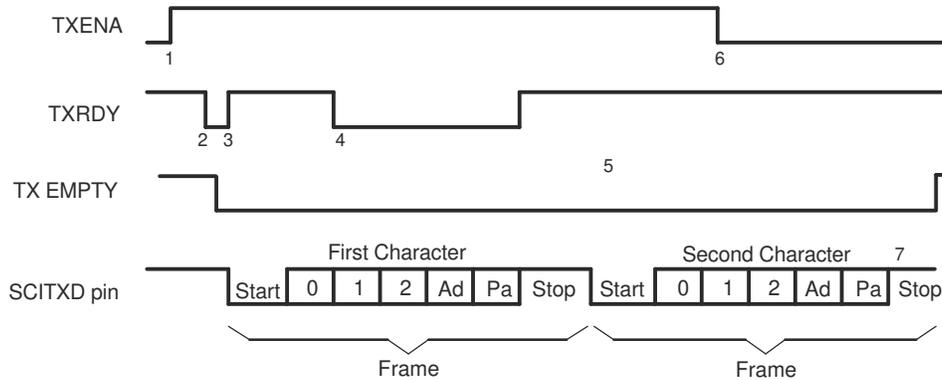
#### Notes:

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



**Figure 18-9. SCI TX Signals in Communications Mode**

**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 that 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 that 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 Interrupts section of the *System Control and Interrupts* 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 9.625 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**

SCI Module Interrupt Reaction Time - Occasional BRKDT or other errors such as FE/PE being triggered can occur if there are tight timings occurring in the application.

Interrupts are not triggered until approximately 7/8 of the stop bit has been detected (approximately 0.875 bit time). Actual value of this delay before ISR entry is:  $((7 * \text{BAUD\_CLK\_PERIOD}) / 8 + 3 * \text{SYSCLK\_PERIOD})$ .

The SCI does not begin reading additional bits/characters until the RX ISR completes, so complete the ISR before the next byte's start bit begins. This leaves approximately 1/8 bit time (approximately 0.125 bit time) to complete the entire ISR, regardless of interrupt cause.

If the ISR is not completed before the beginning of the next start bit (before the RX line goes low again), the SCI module begins reading the start bit late in the wrong location and therefore may read all bits incorrectly until the next correctly aligned start bit (when ISR has sufficient time to process before a start bit again).

Recommended methods for avoiding errors (to accommodate for the 0.875 bit time required for ISR to begin):

1. Keep the RX ISR short. The RX ISR must only be used to move data in the FIFO/buffer into memory where the data can be processed in another, less time-critical function.
  2. Avoid excessive nesting of other interrupts within the SCI RX ISR. Do not allow the nesting to delay SCI RX ISR completion past the approximately 0.125 bit time window allowed.
  3. If additional time (more than the approximately 0.125 bit time) is required, delay can be added in the other device's firmware before transmitting additional data to the C2000 device's SCI RX pin. Delays can be added to the other device as follows:
    - a. Sending bytes with 2 stop bits to the C2000 device provides approximately 1.125 bit time of processing time for C2000 RX ISR to complete.
    - b. Adding manual delay in the firmware of the other device transmitting to the C2000 device after every BYTE transmitted provides (delay + approximately 0.125 bit time) processing time for the C2000 RX ISR to complete.
    - c. Adding manual delay in the firmware of the other device transmitting to the C2000 device after every C2000 RX INTERRUPT occurs provides (delay + approximately 0.125 bit time) processing time for the C2000 RX ISR to complete. This is more difficult to implement because the other device to predict when the transmitted data triggers an RX interrupt on the C2000 device is required. Examples of things that can trigger an RX interrupt are RX-FIFO level being reached, BRKDT being sent, RXERROR occurring, etc.
- 

**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).

---

### 18.11.1 Break Detect

A "break" signal (also called a "break detect" or "break sequence") can be sent to the module to signal to the bus a specific condition. This break signal is defined as a low pulse of a certain amount of time, typically at least 1 packet wide (including a missed stop bit). The SCI has two main methods for detecting a "break" signal sent on the line, with certain limitations for each.

The first for break detect method involves reading the SCIRXST.BRKDT bit. A break condition that triggers the BRKDT bit occurs when the SCI receiver data line (SCIRXD) remains continuously low for at least 9.625 bits, beginning after a missing first stop bit. If the SCIRX line goes high at any point during the 9.625 bits then the SCI does not flag a break detect. To trigger the first stop bit missed, the typical method is to hold the RX line low for:

- 1 start bit
- 8 data bits
- (optional) 1 address bit
- (optional) 1 parity bit
- 1 stop bit
- 9.625 bits of additional time held low

This is a total of 19.625 (systems using no parity or address bit), 20.625 (systems using either parity or address bit but not both), or 21.625 (systems using both parity and address bit) bit times held low.

The second method for break detect is to instead use the SCIRXST.FE, SCIRXST.PE and SCIRXBUF.SAR bits to detect a break signal of 10 or 11 bits of low. ISR code can use the following combination of flags and received data to determine if a break detect occurred:

- Break signal = 11 bits low (requires parity enabled)
  - FE==1
  - PE==1 for odd parity, PE==0 for even parity
  - SCIRXBUF.SAR (received character)==0x00
- Break signal = 10 bits low (requires parity disabled)
  - FE==1
  - SCIRXBUF.SAR (received character)==0x00

### 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. LSPCLK/16 is the maximum baud rate. For example, if LSPCLK is 100MHz, then the maximum baud rate is 6.25Mbps.

**Table 18-3. Asynchronous Baud Register Values for Common SCI Bit Rates**

Baud Rate	LSPCLK Clock Frequency, 100MHz		
	BRR	Actual Baud Rate	% Error
2400	5207 (1457h)	2400	0
4800	2603 (A2Bh)	4800	0
9600	1301 (515h)	9601	0.01
19200	650 (28Ah)	19201	0.01
38400	324 (144h)	38462	0.16

## 18.13 SCI Enhanced Features

The C28x 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 SCI modes 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 the 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; transmit FIFO (TXINT) and receive FIFO (RXINT). The RXINT is the common interrupt for SCI FIFO receive, receive error, and receive FIFO overflow conditions. The TXINT of the standard SCI is disabled and this interrupt serves 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 (SCITXBUF) of the standard SCI functions as a transition buffer before the transmit FIFO and shift register. SCITXBUF is loaded into either the FIFO (when FIFO is enabled) or the TXSHF (when FIFO is disabled). When FIFO is enabled, SCITXBUF loads into the FIFO only after the last bit of the shift register is shifted out, so SCITXBUF cannot be treated as an additional level of buffer. With the FIFO enabled, TXSHF is directly loaded from the FIFO (not TXBUF) after an optional delay value (SCIFFCT). 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 that words in the FIFO are transferred to the transmit shift register is programmable. The SCIFFCT register bits (7–0) FFTXDLY7–FFTXDLY0 define the delay between the word transfer. The delay is defined in the number SCI baud clock cycles. The 8 bit register can define a minimum delay of 0 baud clock cycles and a maximum of 256-baud clock cycles. With zero delay, the SCI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay the SCI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 baud clocks between each words. The programmable delay facilitates communication with slow SCI/UARTs with little CPU intervention.
8. **FIFO status bits.** Both the transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12–8) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO reset the FIFO pointers to zero when these bits are cleared to 0. The FIFOs resumes operation from start once these bits are set to 1.
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 is 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.

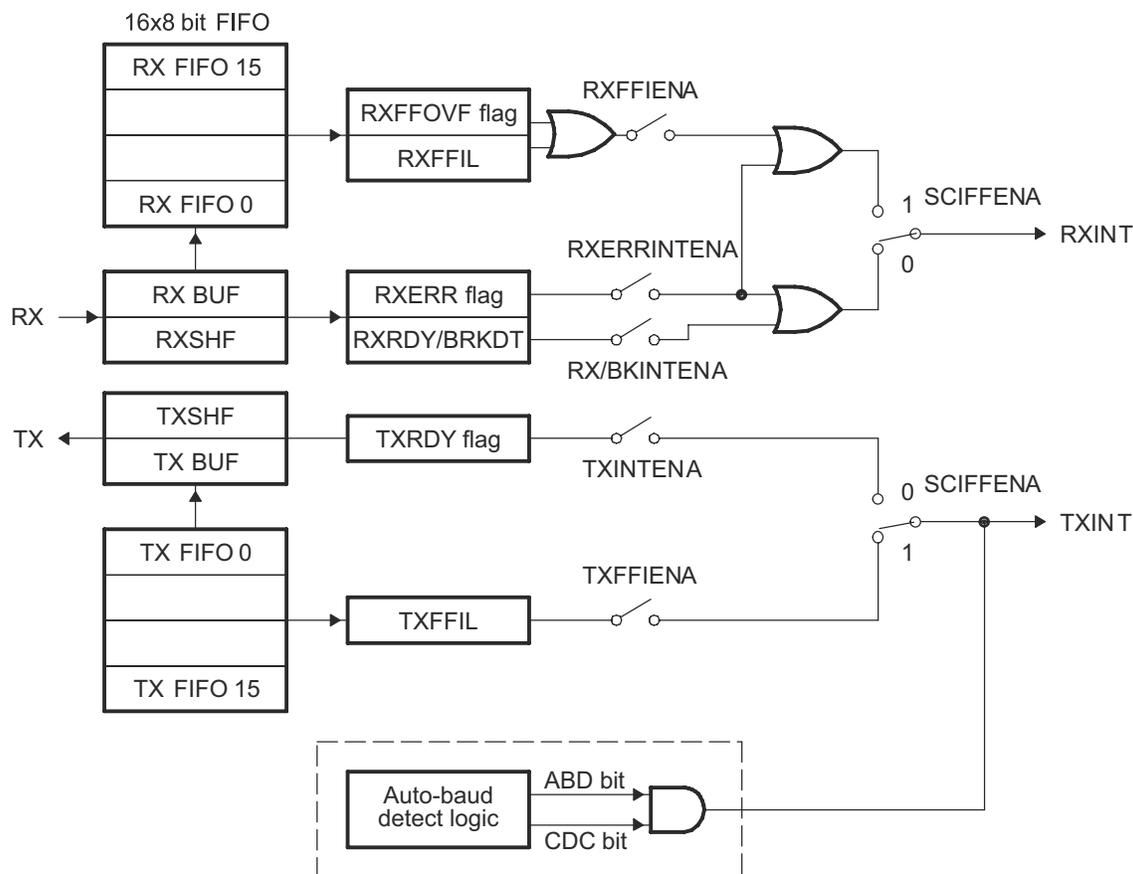


Figure 18-10. SCI FIFO Interrupt Flags and Enable Logic

Table 18-4. SCI Interrupt Flags

FIFO Options <sup>(1)</sup>	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt Line
SCI without FIFO	Receive error	RXERR <sup>(2)</sup>	RXERRINTENA	0	RXINT
	Receive break	BRKDT	RX/BKINTENA	0	RXINT
	Data receive	RXRDY	RX/BKINTENA	0	RXINT
	Transmit empty	TXRDY	TXINTENA	0	TXINT
SCI with FIFO	Receive error and receive break	RXERR	RXERRINTENA	1	RXINT
	FIFO receive	RXFFIL	RXFFIENA	1	RXINT
	Transmit empty	TXFFIL	TXFFIENA	1	TXINT
Auto-baud	Auto-baud detected	ABD	Don't care	x	TXINT

(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 can be enabled to make autobaud logic work.

If ABD is set while CDC is 1, which indicates auto-baud alignment, SCI transmit FIFO interrupt occurs (TXINT). After the interrupt service, the CDC bit must be cleared by software. If CDC remains set even after interrupt service, there can 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 500Kbps.
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 detects the incoming baud rate and sets the ABD bit.
4. The auto-detect hardware updates the baud rate register with the equivalent baud value hex. The logic also generates 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 occurs (TXINT). After the interrupt service, the 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 can work well, this slew rate can 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 C28x SCI boot loader using a lower baud rate.
  - The host can then handshake with the loaded C28x application to set the SCI baud rate register to the desired higher baud rate.
-

## 18.14 Software

### 18.14.1 SCI Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/sci

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 18.14.1.1 Tune Baud Rate via UART Example

FILE: `baud_tune_via_uart.c`

This example demonstrates the process of tuning the UART/SCI baud rate of a C2000 device based on the UART input from another device. As UART does not have a clock signal, reliable communication requires baud rates to be reasonably matched. This example addresses cases where a clock mismatch between devices is greater than is acceptable for communications, requiring baud compensation between boards. As reliable communication only requires matching the EFFECTIVE baud rate, it does not matter which of the two boards executes the tuning (the board with the less-accurate clock source does not need to be the one to tune; as long as one of the two devices tunes to the other, then proper communication can be established).

To tune the baud rate of this device, SCI data (of the desired baud rate) must be sent to this device. The input SCI baud rate must be within the +/- MARGINPERCENT of the TARGETBAUD chosen below. These two variables are defined below, and should be chosen based on the application requirements. Higher MARGINPERCENT will allow more data to be considered "correct" in noisy conditions, and may decrease accuracy. The TARGETBAUD is what was expected to be the baud rate, but due to clock differences, needs to be tuned for better communication robustness with the other device.

NOTE: Lower baud rates have more granularity in register options, and therefore tuning is more affective at these speeds.

*External Connections* for Control Card

- SCIA\_RX/eCAP1 is on GPIO9, connect to incoming SCI communications
- SCIA\_TX is on GPIO8, for observation externally

*Watch Variables*

- *avgBaud* - Baud rate that was detected and set after tuning

#### 18.14.1.2 SCI FIFO Digital Loop Back

FILE: `sci_ex1_loopback.c`

This program uses the internal loop back test mode of the peripheral. Other than boot mode pin configuration, no other hardware configuration is required. The pinmux and SCI modules are configured through the sysconfig file.

This test uses the loopback test mode of the SCI module to send characters starting with 0x00 through 0xFF. The test will send a character and then check the receive buffer for a correct match.

*Watch Variables*

- *loopCount* - Number of characters sent
- *errorCount* - Number of errors detected
- *sendChar* - Character sent
- *receivedChar* - Character received

#### 18.14.1.3 SCI Digital Loop Back with Interrupts

FILE: `sci_ex2_loopback_interrupts.c`

This test uses the internal loop back test mode of the peripheral. Other than boot mode pin configuration, no other hardware configuration is required. Both interrupts and the SCI FIFOs are used.

A stream of data is sent and then compared to the received stream. The SCI-A sent data looks like this:  
00 01

01 02  
02 03

....

FE FF  
FF 00  
etc..

The pattern is repeated forever.

#### Watch Variables

- *sDataA* - Data being sent
- *rDataA* - Data received
- *rDataPointA* - Keep track of where we are in the data stream. This is used to check the incoming data

#### 18.14.1.4 SCI Echoback

FILE: sci\_ex3\_echoback.c

This test receives and echo-backs data through the SCI-A port.

A terminal such as 'putty' can be used to view the data from the SCI and to send information to the SCI. Characters received by the SCI port are sent back to the host.

*Running the Application* Open a COM port with the following settings using a terminal:

- Find correct COM port
- Bits per second = 9600
- Data Bits = 8
- Parity = None
- Stop Bits = 1
- Hardware Control = None

The program will print out a greeting and then ask you to enter a character which it will echo back to the terminal.

#### Watch Variables

- *loopCounter* - the number of characters sent

#### External Connections

Connect the USB cable from Control card J1:A to PC

#### 18.14.1.5 stdout redirect example

FILE: sci\_ex4\_stdout\_redirect.c This test transmits data through the SCI-A port to a terminal

A terminal such as 'putty' can be used to view the data from the SCI. Characters received by the SCI port are sent back to the host.

*Running the Application* Open a COM port with the following settings using a terminal:

- Find correct COM port
- Bits per second = 9600
- Data Bits = 8
- Parity = None
- Stop Bits = 1
- Hardware Control = None

The program will print out three sentences: one to the SCIA, one to CCS, and a final one to SCIA.

#### External Connections

Connect the SCI-A port to a PC via a transceiver and cable.

- *DEVICE\_GPIO\_PIN\_SCIRXDA* is SCI\_A-RXD (Connect to Pin3, PC-TX, of serial DB9 cable)
- *DEVICE\_GPIO\_PIN\_SCITXDA* is SCI\_A-TXD (Connect to Pin2, PC-RX, of serial DB9 cable)

## 18.15 SCI Registers

The section describes the Serial Communication Interface module registers.

### 18.15.1 SCI Base Address Table

**Table 18-5. SCI Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
SciaRegs	SCI_REGS	SCIA_BASE	0x0000_7200	YES
ScibRegs	SCI_REGS	SCIB_BASE	0x0000_7210	YES
ScicRegs	SCI_REGS	SCIC_BASE	0x0000_7220	YES

### 18.15.2 SCI\_REGS Registers

Table 18-6 lists the memory-mapped registers for the SCI\_REGS registers. 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		<a href="#">Go</a>
1h	SCICTL1	Control register 1		<a href="#">Go</a>
2h	SCIHBAUD	Baud rate (high) register		<a href="#">Go</a>
3h	SCILBAUD	Baud rate (low) register		<a href="#">Go</a>
4h	SCICTL2	Control register 2		<a href="#">Go</a>
5h	SCIRXST	Receive status register		<a href="#">Go</a>
6h	SCIRXEMU	Receive emulation buffer register		<a href="#">Go</a>
7h	SCIRXBUF	Receive data buffer		<a href="#">Go</a>
9h	SCITXBUF	Transmit data buffer		<a href="#">Go</a>
Ah	SCIFFTX	FIFO transmit register		<a href="#">Go</a>
Bh	SCIFFRX	FIFO receive register		<a href="#">Go</a>
Ch	SCIFFCT	FIFO control register		<a href="#">Go</a>
Fh	SCIPRI	SCI priority control		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 18-7 shows the codes that are used for access types in this section.

**Table 18-7. SCI\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R -0	Read Returns 0s
Write Type		
W	W	Write
W1S	W 1S	Write 1 to set
Reset or Default Value		
-n		Value after reset or the default value

### 18.15.2.1 SCICCR Register (Offset = 0h) [Reset = 0000h]

SCICCR is shown in [Figure 18-11](#) and described in [Table 18-8](#).

Return to the [Summary Table](#).

SCICCR defines the character format, protocol, and communications mode used by the SCI.

**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_MODE	SCICHAR		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		

**Table 18-8. SCICCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	STOPBITS	R/W	0h	SCI number of stop bits. This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit. Reset type: SYSRSn 0h (R/W) = One stop bit 1h (R/W) = Two stop bits
6	PARITY	R/W	0h	SCI parity odd/even selection. If the PARITY ENABLE bit (SCICCR, bit 5) is set, PARITY (bit 6) designates odd or even parity (odd or even number of bits with the value of 1 in both transmitted and received characters). Reset type: SYSRSn 0h (R/W) = Odd parity 1h (R/W) = Even parity
5	PARITYENA	R/W	0h	SCI parity enable. This bit enables or disables the parity function. If the SCI is in the addressbit multiprocessor mode (set using bit 3 of this register), the address bit is included in the parity calculation (if parity is enabled). For characters of less than eight bits, the remaining unused bits should be masked out of the parity calculation. Reset type: SYSRSn 0h (R/W) = Parity disabled no parity bit is generated during transmission or is expected during reception 1h (R/W) = Parity is enabled
4	LOOPBKENA	R/W	0h	Loop Back test mode enable. This bit enables the Loop Back test mode where the Tx pin is internally connected to the Rx pin. Reset type: SYSRSn 0h (R/W) = Loop Back test mode disabled 1h (R/W) = Loop Back test mode enabled

**Table 18-8. SCICCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3	ADDRIDLE_MODE	R/W	0h	<p>SCI multiprocessor mode control bit.</p> <p>This bit selects one of the multiprocessor protocols. Multiprocessor communication is different from the other communication modes because it uses SLEEP and TXWAKE functions (bits SCICTL1, bit 2 and SCICTL1, bit 3, respectively). The idle-line mode is usually used for normal communications because the address-bit mode adds an extra bit to the frame. The idle-line mode does not add this extra bit and is compatible with RS-232 type communications.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Idle-line mode protocol selected 1h (R/W) = Address-bit mode protocol selected</p>
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>Reset type: SYSRSn</p> <p>0h (R/W) = SCICCHAR_LENGTH_1 1h (R/W) = SCICCHAR_LENGTH_2 2h (R/W) = SCICCHAR_LENGTH_3 3h (R/W) = SCICCHAR_LENGTH_4 4h (R/W) = SCICCHAR_LENGTH_5 5h (R/W) = SCICCHAR_LENGTH_6 6h (R/W) = SCICCHAR_LENGTH_7 7h (R/W) = SCICCHAR_LENGTH_8</p>

### 18.15.2.2 SCICTL1 Register (Offset = 1h) [Reset = 0000h]

SCICTL1 is shown in [Figure 18-12](#) and described in [Table 18-9](#).

Return to the [Summary Table](#).

SCICTL1 controls the receiver/transmitter enable, TXWAKE and SLEEP functions, and the SCI software reset.

**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	RXERRINTENA	SWRESET	RESERVED	TXWAKE	SLEEP	TXENA	RXENA
R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 18-9. 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. Reset type: SYSRSn 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. This reset will not reset the FIFO pointers or flush out the data in TX/RX FIFO. If you need to clear the FIFO then perform SWRESET + TXFFINT + RXFFINT or refer to a channel reset SCIFFTX[SCIRST]. 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 Reset type: SYSRSn 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. There is no time requirement to meet before writing a one to this bit after writing a zero.
4	RESERVED	R	0h	Reserved

**Table 18-9. 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>Reset type: SYSRSn</p> <p>0h (R/W) = Transmit feature is not selected. In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In address-bit mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1</p> <p>1h (R/W) = Transmit feature selected is dependent on the mode, idle-line or address-bit: TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5)</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. The receiver still operates when the SLEEP bit is set however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5-2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Sleep mode disabled</p> <p>1h (R/W) = Sleep mode enabled</p>
1	TXENA	R/W	0h	<p>SCI transmitter enable.</p> <p>Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent. Data written into SCITXBUF when TXENA is disabled will not be transmitted even if the TXENA is enabled later.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmitter disabled</p> <p>1h (R/W) = Transmitter enabled</p>
0	RXENA	R/W	0h	<p>SCI receiver enable.</p> <p>Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers).</p> <p>Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, this will not stop RX errors from triggering interrupts. To disable interrupts from RX errors use the RXERRINTENA bit. To stop propagation of the BRKDT interrupt use the RXBKINTENA bit.</p> <p>The receiver shift register can continue to assemble characters even while RXENA is cleared. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers</p> <p>1h (R/W) = Send received characters to SCIRXEMU and SCIRXBUF</p>

### 18.15.2.3 SCIHBAUD Register (Offset = 2h) [Reset = 0000h]

SCIHBAUD is shown in [Figure 18-13](#) and described in [Table 18-10](#).

Return to the [Summary Table](#).

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**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							

**Table 18-10. SCIHBAUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	BAUD	R/W	0h	SCI 16-bit baud selection Registers SCIHBAUD (MSbyte). 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. $BRR = (SCIHBAUD \ll 8) + (SCILBAUD)$ 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 Reset type: SYSRSn

### 18.15.2.4 SCILBAUD Register (Offset = 3h) [Reset = 0000h]

SCILBAUD is shown in [Figure 18-14](#) and described in [Table 18-11](#).

Return to the [Summary Table](#).

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

**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							

**Table 18-11. SCILBAUD Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	BAUD	R/W	0h	See SCIHBAUD Detailed Description Reset type: SYSRSn

### 18.15.2.5 SCICTL2 Register (Offset = 4h) [Reset = 00C0h]

SCICTL2 is shown in [Figure 18-15](#) and described in [Table 18-12](#).

Return to the [Summary Table](#).

SCICTL2 enables the receive-ready, break-detect, and transmit-ready interrupts as well as transmitter-ready and -empty flags.

**Figure 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-1h	R-1h	R-0h				R/W-0h	R/W-0h

**Table 18-12. SCICTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	TXRDY	R	1h	Transmitter buffer register ready flag. When set, this bit indicates that the transmit data buffer register, SCITXBUF, is ready to receive another character. Writing data to the SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit, TX INT ENA (SCICTL2.0), is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL1.5) or by a system reset. Reset type: SYSRSn 0h (R/W) = SCITXBUF is full 1h (R/W) = SCITXBUF is ready to receive the next character
6	TXEMPTY	R	1h	Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.5), or a system reset, sets this bit. This bit does not cause an interrupt request. Reset type: SYSRSn 0h (R/W) = Transmitter buffer or shift register or both are loaded with data 1h (R/W) = Transmitter buffer and shift registers are both empty
5-2	RESERVED	R	0h	Reserved
1	RXBKINTENA	R/W	0h	Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by either the RXRDY flag or the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BK INT ENA does not prevent the setting of these flags. Reset type: SYSRSn 0h (R/W) = Disable RXRDY/BRKDT interrupt 1h (R/W) = Enable RXRDY/BRKDT interrupt

**Table 18-12. SCICTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	TXINTENA	R/W	0h	<p>SCITXBUF-register interrupt enable.</p> <p>This bit controls the interrupt request caused by the setting of TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set (which indicates SCITXBUF is ready to receive another character).</p> <p>0 Disable TXRDY interrupt 1 Enable TXRDY interrupt.</p> <p>In non-FIFO mode, a dummy (or a valid) data has to be written to SCITXBUF for the first transmit interrupt to occur. This is the case when you enable the transmit interrupt for the first time and also when you re-enable (disable and then enable) the transmit interrupt. If TXINTENA is enabled after writing the data to SCITXBUF, it will not generate an interrupt.</p> <p>Reset type: SYSRSn 0h (R/W) = Disable TXRDY interrupt 1h (R/W) = Enable TXRDY interrupt</p>

### 18.15.2.6 SCIRXST Register (Offset = 5h) [Reset = 0000h]

SCIRXST is shown in [Figure 18-16](#) and described in [Table 18-13](#).

Return to the [Summary Table](#).

SCIRXST contains seven bits that are receiver status flags (two of which can generate interrupt requests). Each time a complete character is transferred to the receiver buffers (SCIRXEMU and SCIRXBUF), the status flags are updated.

**Figure 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

**Table 18-13. SCIRXST Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved

**Table 18-13. SCIRXST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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, channel reset (SCIRST), or by a system reset.</p> <p>Note: SCI Module Interrupt Reaction Time - Occasional BRKDT or other errors such as FE/PE being triggered could occur if there are tight timings occurring in the application.</p> <p>Interrupts are not triggered until approximately 7/8 of the stop bit has been detected (~0.875 bit time). Actual value of this delay before ISR entry is: <math>((7 * \text{BAUD\_CLK\_PERIOD}) / 8 + 3 * \text{SYSCLK\_PERIOD})</math>.</p> <p>The SCI will not begin reading additional bits/characters until the RX ISR completes so it is critical to complete the ISR before the next byte's start bit begins. This leaves approximately 1/8 bit time (~0.125 bit time) to complete the entire ISR, regardless of interrupt cause. If the ISR is not completed before the beginning of the next start bit (before the RX line goes low again), the SCI module will begin reading the start bit late in the wrong location and therefore may read all bits incorrectly until the next correctly aligned start bit (when ISR has sufficient time to process before a start bit again).</p> <p>Recommended methods for avoiding errors (to accommodate for the 0.875 bit time required for ISR to begin):</p> <ol style="list-style-type: none"> <li>1. Keep the RX ISR short. The RX ISR should only be used to move data in the FIFO/buffer into memory where it can be processed in another, less time-critical function.</li> <li>2. Avoid excessive nesting of other interrupts within the SCI RX ISR. Do not allow the nesting to delay SCI RX ISR completion past the ~0.125 bit time window allowed.</li> <li>3. If additional time (more than the ~0.125 bit time) is required, delay can be added in the other device's firmware before transmitting additional data to the C2000 device's SCI RX pin. Delays can be added to the other device as follows: <ul style="list-style-type: none"> <li>A. Sending bytes with 2 stop bits to the C2000 device provides ~1.125 bit time of processing time for C2000 RX ISR to complete.</li> <li>B. Adding manual delay in the firmware of the other device transmitting to the C2000 device after every BYTE transmitted will provide (delay + ~0.125 bit time) processing time for the C2000 RX ISR to complete.</li> <li>C. Adding manual delay in the firmware of the other device transmitting to the C2000 device after every C2000 RX INTERRUPT occurs will provide (delay + ~0.125 bit time) processing time for the C2000 RX ISR to complete. This is more difficult to implement as it requires the other device to predict when its transmitted data will trigger an RX interrupt on the C2000 device. Examples of things that can trigger an RX interrupt are RX-FIFO level being reached, BRKDT being sent, RXERROR occurring, etc.</li> </ul> </li> </ol> <p>Reset type: SYSRSn  0h (R/W) = No error flags set  1h (R/W) = Error flag(s) set</p>
6	RXRDY	R	0h	<p>SCI receiver-ready flag.</p> <p>When a new character is ready to be read from the SCIRXBUF register, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by a reading of the SCIRXBUF register, by an active SW RESET, channel reset (SCIRST), or by a system reset.</p> <p>Reset type: SYSRSn  0h (R/W) = No new character in SCIRXBUF  1h (R/W) = Character ready to be read from SCIRXBUF</p>

**Table 18-13. SCIRXST Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
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 9.625 bits, beginning after a missing first stop bit. If the SCIRX line goes high at any point during the 9.625 bits then the SCI will not flag a break detect. In order to trigger the first stop bit missed, the typical method is to hold the RX line low for 1 start bit, 8 data bits, 1 optional address bit, 1 optional parity bit, 1 stop bit, and 9.625 bits of additional time held low. This is a total of 19.625 (no parity/address bit), 20.625 (either parity or address bit), or 21.625 (both parity and address bit) bit times.</p> <p>To instead detect a 'break seq' or 'break sequence' of 11 bits of low voltage level (0), ISR code can use the following combination of flags and received data: FE==1 &amp;&amp; PE==1 &amp;&amp; SCIRXBUF.SAR (received character)==0x00. This assumes parity enabled and odd parity set. With even parity, PE==0 instead. The detection of 11 bits of low/0 can be reduced to 10 bits of low if no parity bit is used (then PE flag does not matter to detect the sequence).</p> <p>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.</p> <p>A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1.</p> <p>BRKDT is cleared by an active SW RESET, SCIRST bit, or by a system reset. It is not cleared by receipt of a character after the break is detected.</p> <p>If Break Detect (BRKDT) is set, then RXRDY won't be set and there will be no further interrupts after the first interrupt where there is an error detected if a SW reset, channel reset, or system reset is not performed. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit, channel reset (SCIRST), or by a system reset.</p> <p>NOTE: If your system is susceptible to break detects, ensure that you have a pull-up resistor on the SCI-RX pin to provide proper return-to-high signal behavior and noise immunity.</p> <p>NOTE: To monitor a break detect, place an oscilloscope on the C2000 SCI-RX line and monitor for a low-signal greater than 9.625 bits wide. If this is found and a break is not expected, please correct the software in the other device that is transmitting to this C2000 device. There should never be a low-signal greater than 9.625 bits wide on the SCI-RX line of the C2000 device unless a break detect is being transmitted purposely.</p> <p>Reset type: SYSRSn                      0h (R/W) = No break condition                      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, channel reset (SCIRST), or by a system reset. NOTE: FE will be flagged prior to BRKDT, except when RX is in sleep mode. In sleep mode, when there is no RX WAKEUP and RXD line is low for greater than 10 bits, BRKDT will be flagged while FE will not be flagged.</p> <p>Reset type: SYSRSn                      0h (R/W) = No framing error detected                      1h (R/W) = Framing error detected</p>

**Table 18-13. 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, channel reset (SCIRST), or a system reset.</p> <p>Reset type: SYSRSn</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, channel reset (SCIRST), or a system reset.</p> <p>Reset type: SYSRSn</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>Reset type: SYSRSn</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"> <li>- The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode)</li> <li>- The reading of SCIRXBUF</li> <li>- An active SW RESET</li> <li>- Channel reset (SCIRST)</li> <li>- A system reset</li> </ul>
0	RESERVED	R	0h	Reserved

### 18.15.2.7 SCIRXEMU Register (Offset = 6h) [Reset = 0000h]

SCIRXEMU is shown in [Figure 18-17](#) and described in [Table 18-14](#).

Return to the [Summary Table](#).

Normal SCI data-receive operations read the data received from the SCIRXBUF register. The SCIRXEMU register is used principally by the emulator (EMU) because it can continuously read the data received for screen updates without clearing the RXRDY flag. SCIRXEMU is cleared by a system reset. This is the register that should be used in an emulator watch window to view the contents of the SCIRXBUF register. SCIRXEMU is not physically implemented

it is just a different address location to access the SCIRXBUF register without clearing the RXRDY flag.

**Figure 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							

**Table 18-14. SCIRXEMU Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	ERXDT	R	0h	Receive emulation buffer data Reset type: SYSRSn

### 18.15.2.8 SCIRXBUF Register (Offset = 7h) [Reset = 0000h]

SCIRXBUF is shown in [Figure 18-18](#) and described in [Table 18-15](#).

Return to the [Summary Table](#).

When the current data received is shifted from RXSHF to the receiver buffer, flag bit RXRDY is set and the data is ready to be read. If the RXBKINTENA bit (SCICTL2.1) is set, this shift also causes an interrupt. When SCIRXBUF is read, the RXRDY flag is reset. SCIRXBUF is cleared by a system reset.

**Figure 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							

**Table 18-15. SCIRXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SCIFFFE	R	0h	SCIFFFE. SCI FIFO Framing error flag bit (applicable only if the FIFO is enabled) Note: 'SCIFFFE' is meant to serve as a flag for the specific set of data being received/read in the SCIRXBUF register. Each set of data received into the FIFO will have this information. The 'FE' bit within the SCIRXST register can be thought off as high level error flag where the flag will get set if any data that has been received has a framing error. Reset type: SYSRSn 0h (R/W) = No frame error occurred while receiving the character, in bits 7-0. This bit is associated with the character on the top of the FIFO. 1h (R/W) = A frame error occurred while receiving the character in bits 7-0. This bit is associated with the character on the top of the FIFO.
14	SCIFFPE	R	0h	SCIFFPE. SCI FIFO parity error flag bit (applicable only if the FIFO is enabled) Note: 'SCIFFPE' is meant to serve as a flag for the specific set of data being received/read in the SCIRXBUF register. Each set of data received into the FIFO will have this information. The 'PE' bit within the SCIRXST register can be thought off as high level error flag where the flag will get set if any data that has been received has a parity error. Note: If the parity is changed in the middle of data reception, the SCI module will not reinterpret the data with the new parity or other settings that may have changed. Therefore, changing the parameter, the FIFO should be cleared or the user should acknowledge that there will most likely be errors in the data caused by the change. Note: If RX parity errors are occurring intermittently this could be due to the length of the SCI ISR. To help prevent this, ensure that interrupt nesting is limited, increase the SCI interrupt priority, and move as much of the processing as possible out of the ISR (to reduce ISR time to the absolute minimum). Reset type: SYSRSn 0h (R/W) = No parity error occurred while receiving the character, in bits 7-0. This bit is associated with the character on the top of the FIFO. 1h (R/W) = A parity error occurred while receiving the character in bits 7-0. This bit is associated with the character on the top of the FIFO.
13-8	RESERVED	R	0h	Reserved

**Table 18-15. SCIRXBUF Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-0	SAR	R	0h	Receive Character bits Reset type: SYSRSn

### 18.15.2.9 SCITXBUF Register (Offset = 9h) [Reset = 0000h]

SCITXBUF is shown in [Figure 18-19](#) and described in [Table 18-16](#).

Return to the [Summary Table](#).

Data bits to be transmitted are written to SCITXBUF. These bits must be rightjustified because the leftmost bits are ignored for characters less than eight bits long. The transfer of data from this register to the TXSHF transmitter shift register sets the TXRDY flag (SCICTL2.7), indicating that SCITXBUF is ready to receive another set of data. If bit TXINTENA (SCICTL2.0) is set, this data transfer also causes an interrupt.

**Figure 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							

**Table 18-16. SCITXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	TXDT	R/W	0h	Transmit data buffer Reset type: SYSRSn

### 18.15.2.10 SCIFFTX Register (Offset = Ah) [Reset = A000h]

SCIFFTX is shown in [Figure 18-20](#) and described in [Table 18-17](#).

Return to the [Summary Table](#).

SCIFFTX controls the transmit FIFO interrupt, FIFO enhancements, and reset for the SCI transmit and receive channels.

**Figure 18-20. SCIFFTX Register**

15		14		13		12		11		10		9		8	
SCIRST		SCIFFENA		TXFIFORESET						TXFFST					
R/W-1h		R/W-0h		R/W-1h						R-0h					
7		6		5		4		3		2		1		0	
TXFFINT		TXFFINTCLR		TXFFIENA						TXFFIL					
R-0h		R-0/W1S-0h		R/W-0h						R/W-0h					

**Table 18-17. SCIFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SCIRST	R/W	1h	SCI Reset 0 A write of 0 will cause a SW RESET + a RESET of TXFFINT and RXFFINT, essentially clearing TX/RX FIFO content. The SCI will be held in reset until a write of 1. Additionally it resets the RXFFOVF, PE, OE, FE, RXERROR, BRKDET, RXRDY, and RXWAKE flags. It will also set TXRDY and TXEMPTY bits as 1. 1 SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work. Reset type: SYSRSn
14	SCIFFENA	R/W	0h	SCI FIFO enable Reset type: SYSRSn 0h (R/W) = SCI FIFO enhancements are disabled 1h (R/W) = SCI FIFO enhancements are enabled
13	TXFIFORESET	R/W	1h	Transmit FIFO reset Reset type: SYSRSn 0h (R/W) = Reset the FIFO pointer to zero and hold in reset 1h (R/W) = Re-enable transmit FIFO operation
12-8	TXFFST	R	0h	FIFO status Reset type: SYSRSn 0h (R/W) = Transmit FIFO is empty 1h (R/W) = Transmit FIFO has 1 words 2h (R/W) = Transmit FIFO has 2 words 3h (R/W) = Transmit FIFO has 3 words 4h (R/W) = Transmit FIFO has 4 words 5h (R/W) = Transmit FIFO has 5 words 6h (R/W) = Transmit FIFO has 6 words 7h (R/W) = Transmit FIFO has 7 words 8h (R/W) = Transmit FIFO has 8 words 9h (R/W) = Transmit FIFO has 9 words Ah (R/W) = Transmit FIFO has 10 words Bh (R/W) = Transmit FIFO has 11 words Ch (R/W) = Transmit FIFO has 12 words Dh (R/W) = Transmit FIFO has 13 words Eh (R/W) = Transmit FIFO has 14 words Fh (R/W) = Transmit FIFO has 15 words 10h (R/W) = Transmit FIFO has 16 words
7	TXFFINT	R	0h	Transmit FIFO interrupt Reset type: SYSRSn 0h (R/W) = TXFIFO interrupt has not occurred, read-only bit 1h (R/W) = TXFIFO interrupt has occurred, read-only bit

**Table 18-17. SCIFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	TXFFINTCLR	R-0/W1S	0h	Transmit FIFO clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero 1h (R/W) = Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA	R/W	0h	Transmit FIFO interrupt enable Reset type: SYSRSn 0h (R/W) = TX FIFO interrupt is disabled 1h (R/W) = TX FIFO interrupt is enabled. This interrupt is triggered whenever the transmit FIFO status (TXFFST) bits match (equal to or less than) the interrupt trigger level bits TXFFIL (bits 4-0).
4-0	TXFFIL	R/W	0h	TXFFIL4-0 Transmit FIFO interrupt level bits. The transmit FIFO generates an interrupt whenever the FIFO status bits (TXFFST4-0) are less than or equal to the FIFO level bits (TXFFIL4-0). The maximum value that can be assigned to these bits to generate an interrupt cannot be more than the depth of the TX FIFO. The default value of these bits after reset is 00000b. Users should set TXFFIL to best fit their application needs by weighing between the CPU overhead to service the ISR and the best possible usage of SCI bus bandwidth. Reset type: SYSRSn

### 18.15.2.11 SCIFFRX Register (Offset = Bh) [Reset = 201Fh]

SCIFFRX is shown in [Figure 18-21](#) and described in [Table 18-18](#).

Return to the [Summary Table](#).

SCIFFRX controls the receive FIFO interrupt, receive FIFO reset, and status of the receive FIFO overflow.

**Figure 18-21. SCIFFRX Register**

15	14	13	12	11	10	9	8	
RXFFOVF	RXFFOVRCLR	RXFIFORESET	RXFFST					
R-0h	R-0/W1S-0h	R/W-1h	R-0h					
7	6	5	4	3	2	1	0	
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL					
R-0h	W-0h	R/W-0h	R/W-1Fh					

**Table 18-18. SCIFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RXFFOVF	R	0h	Receive FIFO overflow. This will function as flag, but cannot generate interrupt by itself. This condition will occur while receive interrupt is active. Receive interrupts should service this flag condition. This bit is cleared by RXFFOVRCLR, a channel reset (SCIRST), or a system reset. Reset type: SYSRSn 0h (R/W) = Receive FIFO has not overflowed, read-only bit 1h (R/W) = Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost
14	RXFFOVRCLR	R-0/W1S	0h	RXFFOVF clear Note: Both RXFFIL and RXFFOVF flags are ORed together, so they need to be cleared at the same time (RXFFINTCLR & RXFFOVRCLR) during overflow scenarios else it will prevent further interrupts from occurring. Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on RXFFOVF flag bit, Bit reads back a zero 1h (R/W) = Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFORESET	R/W	1h	Receive FIFO reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Re-enable receive FIFO operation

**Table 18-18. SCIFFRX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12-8	RXFFST	R	0h	<p>FIFO status</p> <p>Reset type: SYSRSn</p> <p>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</p>
7	RXFFINT	R	0h	<p>Receive FIFO interrupt</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = RXFIFO interrupt has not occurred, read-only bit            1h (R/W) = RXFIFO interrupt has occurred, read-only bit</p>
6	RXFFINTCLR	W	0h	<p>Receive FIFO interrupt clear</p> <p>Note: Both RXFFIL and RXFFOVF flags are ORed together, so they need to be cleared at the same time (RXFFINTCLR &amp; RXFFOVRCLR) during overflow scenarios else it will prevent further interrupts from occurring.</p> <p>Reset type: SYSRSn</p> <p>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</p>
5	RXFFIENA	R/W	0h	<p>Receive FIFO interrupt enable</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = RX FIFO interrupt is disabled            1h (R/W) = RX FIFO interrupt is enabled. This interrupt is triggered whenever the receive FIFO status (RXFFST) bits match (equal to or greater than) the interrupt trigger level bits RXFFIL (bits 4-0).</p>
4-0	RXFFIL	R/W	1Fh	<p>Receive FIFO interrupt level bits</p> <p>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 1111b. 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.</p> <p>Reset type: SYSRSn</p>

### 18.15.2.12 SCIFFCT Register (Offset = Ch) [Reset = 0000h]

SCIFFCT is shown in [Figure 18-22](#) and described in [Table 18-19](#).

Return to the [Summary Table](#).

SCIFFCT contains the status of auto-baud detect, clears the auto-baud flag, and calibrate for A-detect bit.

**Figure 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							

**Table 18-19. SCIFFCT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	ABD	R	0h	Auto-baud detect (ABD) bit Reset type: SYSRSn 0h (R/W) = Auto-baud detection is not complete. 'A','a' character has not been received successfully. 1h (R/W) = Auto-baud hardware has detected 'A' or 'a' character on the SCI receive register. Auto-detect is complete.
14	ABDCLR	W	0h	ABD-clear bit Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on ABD flag bit. Bit reads back a zero. 1h (R/W) = Write 1 to clear ABD flag in bit 15.
13	CDC	R/W	0h	CDC calibrate A-detect bit Reset type: SYSRSn 0h (R/W) = Disables auto-baud alignment 1h (R/W) = Enables auto-baud alignment
12-8	RESERVED	R	0h	Reserved
7-0	FFTXDLY	R/W	0h	FIFO transfer delay. These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in the number of SCI serial baud clock cycles. The 8 bit register could define a minimum delay of 0 baud clock cycles and a maximum of 256 baud clock cycles In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In FIFO mode, TXBUF should not be treated as one additional level of buffer. The delayed transmit feature will help to create an auto-flow scheme without RTS/CTS controls as in standard UARTS. When SCI is configured for one stop-bit, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to. When SCI is configured for two stop-bits, delay introduced by FFTXDLY between one frame and the next frame is equal to number of baud clock cycles that FFTXDLY is set to minus 1. Reset type: SYSRSn

### 18.15.2.13 SCIPRI Register (Offset = Fh) [Reset = 0000h]

SCIPRI is shown in [Figure 18-23](#) and described in [Table 18-20](#).

Return to the [Summary Table](#).

SCIPRI determines what happens when an emulation suspend event occurs.

**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	

**Table 18-20. SCIPRI Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-5	RESERVED	R	0h	Reserved
4-3	FREESOFT	R/W	0h	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. Reset type: SYSRSn 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

### 18.15.3 SCI Registers to Driverlib Functions

**Table 18-21. SCI Registers to Driverlib Functions**

File	Driverlib Function
<b>SCICCR</b>	
sci.c	SCI_setConfig
sci.h	SCI_setParityMode
sci.h	SCI_getParityMode
sci.h	SCI_setAddrMultiProcessorMode
sci.h	SCI_setIdleMultiProcessorMode
sci.h	SCI_getConfig
sci.h	SCI_enableLoopback
sci.h	SCI_disableLoopback
<b>SCICTL1</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_setWakeFlag
sci.h	SCI_enableModule
sci.h	SCI_disableModule
sci.h	SCI_enableTxModule
sci.h	SCI_disableTxModule

**Table 18-21. SCI Registers to Driverlib Functions (continued)**

File	Driverlib Function
sci.h	SCI_enableRxModule
sci.h	SCI_disableRxModule
sci.h	SCI_enableSleepMode
sci.h	SCI_disableSleepMode
sci.h	SCI_performSoftwareReset
<b>SCIHBAUD</b>	
sci.c	SCI_setConfig
sci.c	SCI_setBaud
sci.h	SCI_lockAutobaud
sci.h	SCI_getConfig
<b>SCILBAUD</b>	
sci.c	SCI_setConfig
sci.c	SCI_setBaud
sci.h	SCI_lockAutobaud
sci.h	SCI_getConfig
<b>SCICTL2</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.h	SCI_isSpaceAvailableNonFIFO
sci.h	SCI_isTransmitterBusy
<b>SCIRXST</b>	
sci.c	SCI_getInterruptStatus
sci.h	SCI_isDataAvailableNonFIFO
sci.h	SCI_getRxStatus
<b>SCIRXEMU</b>	
-	
<b>SCIRXBUF</b>	
sci.c	SCI_readCharArray
sci.h	SCI_readCharBlockingFIFO
sci.h	SCI_readCharBlockingNonFIFO
sci.h	SCI_readCharNonBlocking
<b>SCITXBUF</b>	
sci.c	SCI_writeCharArray
sci.h	SCI_writeCharBlockingFIFO
sci.h	SCI_writeCharBlockingNonFIFO
sci.h	SCI_writeCharNonBlocking
<b>SCIFFTX</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.c	SCI_clearInterruptStatus
sci.h	SCI_setFIFOInterruptLevel
sci.h	SCI_getFIFOInterruptLevel
sci.h	SCI_disableModule

**Table 18-21. SCI Registers to Driverlib Functions (continued)**

File	Driverlib Function
sci.h	SCI_enableFIFO
sci.h	SCI_disableFIFO
sci.h	SCI_isFIFOEnabled
sci.h	SCI_resetTxFIFO
sci.h	SCI_resetChannels
sci.h	SCI_getTxFIFOStatus
sci.h	SCI_isTransmitterBusy
<b>SCIFFRX</b>	
sci.c	SCI_enableInterrupt
sci.c	SCI_disableInterrupt
sci.c	SCI_getInterruptStatus
sci.c	SCI_clearInterruptStatus
sci.h	SCI_setFIFOInterruptLevel
sci.h	SCI_getFIFOInterruptLevel
sci.h	SCI_enableFIFO
sci.h	SCI_resetRxFIFO
sci.h	SCI_getRxFIFOStatus
sci.h	SCI_getOverflowStatus
sci.h	SCI_clearOverflowStatus
<b>SCIFFCT</b>	
sci.h	SCI_lockAutobaud
<b>SCIPRI</b>	
-	

This page intentionally left blank.

Chapter 19  
**Serial Peripheral Interface (SPI)**

---



This chapter describes the serial peripheral interface (SPI) which is a high-speed synchronous serial input and output (I/O) port that allows a serial bit stream of programmed length (one to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the MCU controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion using 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.

<b>19.1 Introduction</b> .....	<b>1832</b>
<b>19.2 System-Level Integration</b> .....	<b>1834</b>
<b>19.3 SPI Operation</b> .....	<b>1837</b>
<b>19.4 Programming Procedure</b> .....	<b>1847</b>
<b>19.5 Software</b> .....	<b>1853</b>
<b>19.6 SPI Registers</b> .....	<b>1854</b>

## 19.1 Introduction

### 19.1.1 Features

The SPI module features include:

- SPISOMI: SPI slave-output/master-input pin
- SPISIMO: SPI slave-input/master-output pin
- SPIS $\overline{T}$ E: 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 data sheet for more details.
- Data word length: 1 to 16 data bits
- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt- driven or polled algorithm
- 16-level transmit/receive FIFO
- High-speed mode
- Delayed transmit control
- 3-wire SPI mode
- SPIS $\overline{T}$ E inversion for digital audio interface receive mode on devices with two SPI modules

### 19.1.2 SPI Related Collateral

#### Foundational Materials

- [C2000 Academy - SPI](#)
- [KeyStone Architecture Serial Peripheral Interface \(SPI\)](#)

#### Getting Started Materials

- [SPI: Microcontroller overview](#) (Video)

### 19.1.3 Block Diagram

Figure 19-1 shows the SPI CPU interfaces.

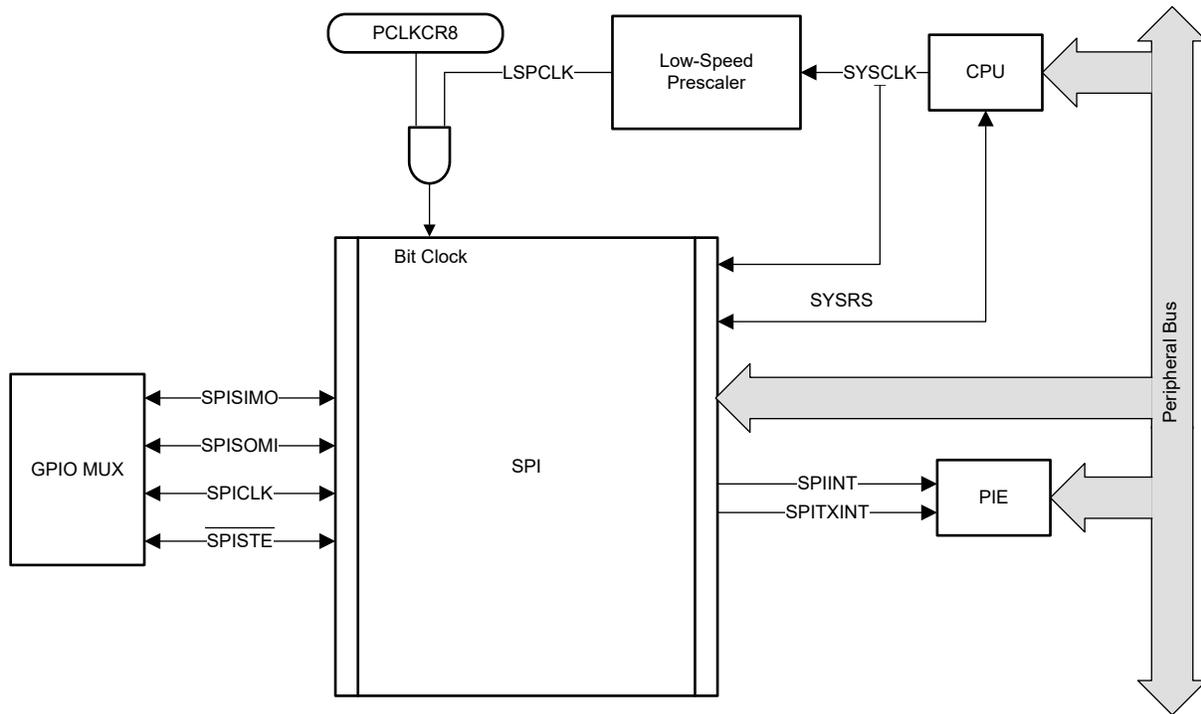


Figure 19-1. SPI CPU Interface

## 19.2 System-Level Integration

This section describes the various functionality that is applicable to the device integration. These features require configuration of other modules in the device that are not within the scope of this chapter.

### 19.2.1 SPI Module Signals

[Table 19-1](#) classifies and provides a summary of the SPI module signals.

**Table 19-1. SPI Module Signal Summary**

Signal Name	Description
<b>External Signals</b>	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
<b>Control</b>	
SPI Clock Rate	LSPCLK
<b>Interrupt Signals</b>	
SPIINT/SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPIINT) Receive interrupt in FIFO mode
SPITXINT	Transmit interrupt in FIFO mode

### Special Considerations

The  $\overline{\text{SPISTE}}$  signal provides the ability to gate any spurious clock and data pulses when the SPI is in slave mode. A HIGH logic signal on  $\overline{\text{SPISTE}}$  does not allow the slave to receive data. This prevents the SPI slave from losing synchronization with the master. TI does not recommend that the  $\overline{\text{SPISTE}}$  always be tied to the active state.

If the SPI slave does ever lose synchronization with the master, toggling SPISWRESET resets the internal bit counter as well as the various status flags in the module. By resetting the bit counter, the SPI interprets the next clock transition as the first bit of a new transmission. The register bit fields that are reset by SPISWRESET are found in [Section 19.6](#).

### Configuring a GPIO to Emulate $\overline{\text{SPISTE}}$

In many systems, a SPI master can be connected to multiple SPI slaves using multiple instances of  $\overline{\text{SPISTE}}$ . Though this SPI module does not natively support multiple  $\overline{\text{SPISTE}}$  signals, it is possible to emulate this behavior in software using GPIOs. In this configuration, the SPI must be configured as the master. Rather than using the GPIO Mux to select  $\overline{\text{SPISTE}}$ , the application can configure pins to be GPIO outputs, one GPIO per SPI slave. Before transmitting any data, the application can drive the desired GPIO to the active state. Immediately after the transmission has been completed, the GPIO chip select can be driven to the inactive state. This process can be repeated for many slaves that share the SPICLK, SPISIMO, and SPISOMI lines.

## 19.2.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the 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 must 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 *General-Purpose Input/Output (GPIO)* chapter for more details on GPIO mux and settings.

### 19.2.2.1 GPIOs Required for High-Speed Mode

The high-speed mode of the SPI is available on all GPIO mux options. To enable the high-speed enhancements, set SPICCR.HS\_MODE to 1. Make sure that the capacitive loading on the pin does not exceed the value stated in the device data sheet.

When not operating in high-speed mode or if the capacitive loading on the pins exceed the value stated in the device data sheet, SPICCR.HS\_MODE can be set to 0.

## 19.2.3 SPI Interrupts

This section includes information on the available interrupts present in the SPI module. The SPI module contains two interrupt lines: SPIINT/SPIRXINT and SPITXINT. When the SPI is operating in non-FIFO mode, all available interrupts are routed together to generate the single SPIINT interrupt. When FIFO mode is used, both SPIRXINT and SPITXINT can be generated.

### SPIINT/SPIRXINT

When the SPI is operating in non-FIFO mode, the interrupt generated is called SPIINT. If FIFO enhancements are enabled, the interrupt is called SPIRXINT. These interrupts share the same interrupt vector in the Peripheral Interrupt Expansion (PIE) block.

In non-FIFO mode, two conditions can trigger an interrupt: a transmission is complete (INT\_FLAG), or there is overrun in the receiver (OVERRUN\_FLAG). Both of these conditions share the same interrupt vector: SPIINT.

The transmission complete flag (INT\_FLAG) indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced. At the same time this bit is set, the received character is placed in the receiver buffer (SPIRXBUF). The INT\_FLAG generates an interrupt on the SPIINT vector if the SPIINTENA bit is set.

The receiver overrun flag (OVERRUN\_FLAG) indicates that a transmit or receive operation has completed before the previous character has been read from the buffer. The OVERRUN\_FLAG generates an interrupt on the SPIINT vector if the OVERRUNINTENA bit is set and OVERRUN\_FLAG was previously cleared.

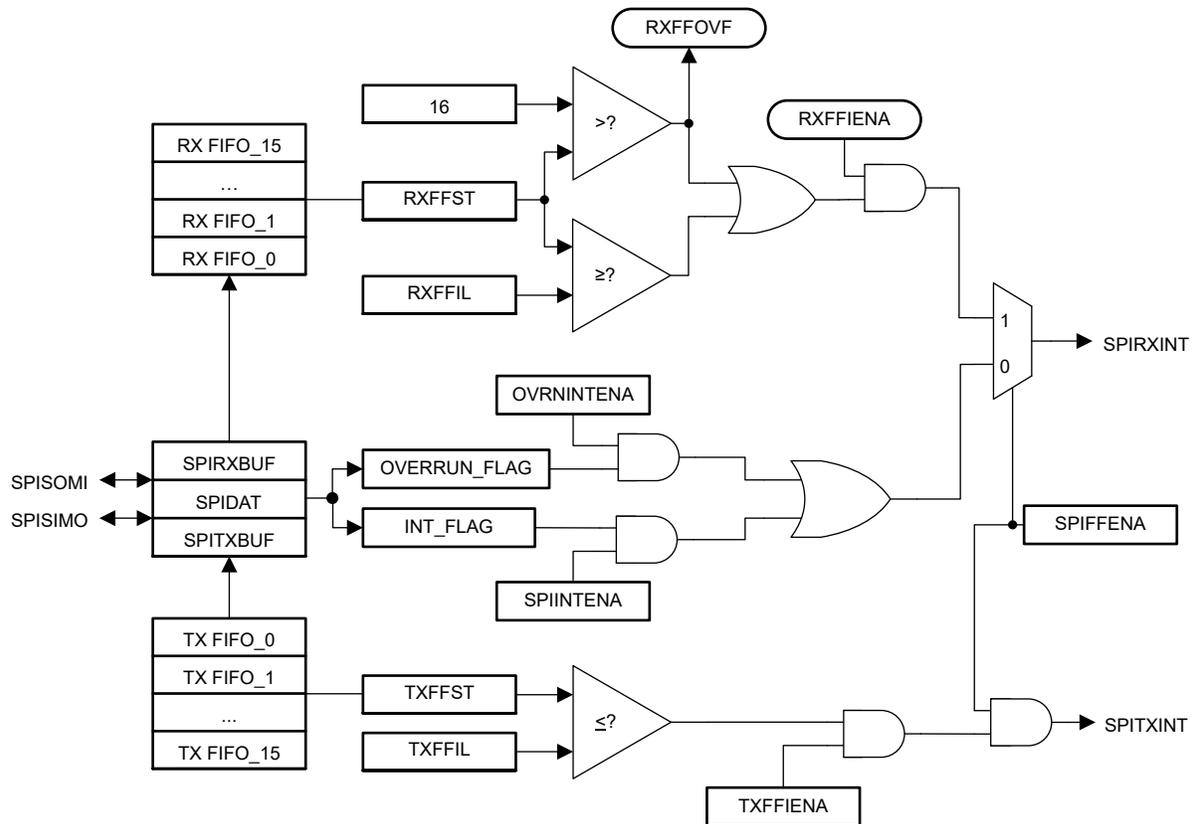
In FIFO mode, the SPI can interrupt the CPU upon a match condition between the current receive FIFO status (RXFFST) and the receive FIFO interrupt level (RXFFIL). If RXFFST is greater than or equal to RXFFIL, the receive FIFO interrupt flag (RXFFINT) is set. SPIRXINT is triggered in the PIE block, if RXFFINT is set and the receive FIFO interrupt is enabled (RXFFIENA = 1).

### SPITXINT

The SPITXINT interrupt is not available when the SPI is operating in non-FIFO mode.

In FIFO mode, the SPITXINT behavior is similar to the SPIRXINT. SPITXINT is generated upon a match condition between the current transmit FIFO status (TXFFST) and the transmit FIFO interrupt level (TXFFIL). If TXFFST is less than or equal to TXFFIL, the transmit FIFO interrupt flag (TXFFINT) is set. SPITXINT is triggered in the PIE block, if TXFFINT is set and the transmit FIFO interrupt is enabled in the SPI module (TXFFIENA = 1).

[Figure 19-2](#) and [Table 19-2](#) show how these control bits influence the SPI interrupt generation.



**Figure 19-2. SPI Interrupt Flags and Enable Logic Generation**

**Table 19-2. SPI Interrupt Flag Modes**

FIFO Options	SPI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable (SPIFFENA)	Interrupt Line <sup>(1)</sup>
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 C28x devices.

### 19.3 SPI Operation

This section describes the various modes of operation of the SPI. Included are explanations of the operational modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

#### 19.3.1 Introduction to Operation

Figure 19-3 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master transfers data by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLK\_PHASE bit is high, data is transmitted and received a half-cycle before the SPICLK transition. As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because the master controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

The SPI operates in master or slave mode. The MASTER\_SLAVE bit selects the operating mode and the source of the SPICLK signal.

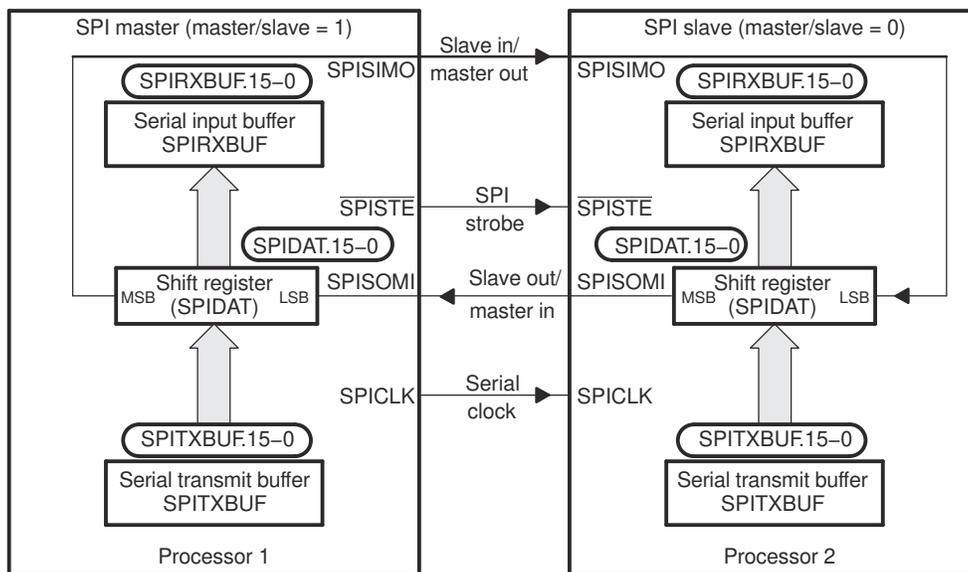


Figure 19-3. SPI Master/Slave Connection

### 19.3.2 Master Mode

In master mode (MASTER\_SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR register determines both the transmit and receive bit transfer rate for the network. SPIBRR can select 125 different data transfer rates.

Data written to SPIDAT or SPITXBUF initiates data transmission on the SPISIMO pin, MSB (most-significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least-significant bit) of SPIDAT. When the selected number of bits has been transmitted, the received data is transferred to the SPIRXBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIRXBUF.

When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- SPIDAT contents are transferred to SPIRXBUF.
- INT\_FLAG bit is set to 1.
- If there is valid data in the transmit buffer SPITXBUF, as indicated by the transmit buffer full flag (BUFFULL\_FLAG), this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- If the SPIINTENA bit is set to 1, an interrupt is asserted.

In a typical application, the  $\overline{\text{SPISTE}}$  pin serves as a chip-enable pin for a SPI slave 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 19-4](#) is a block diagram of the SPI in master mode. The block diagram shows the basic control blocks available in SPI master mode.

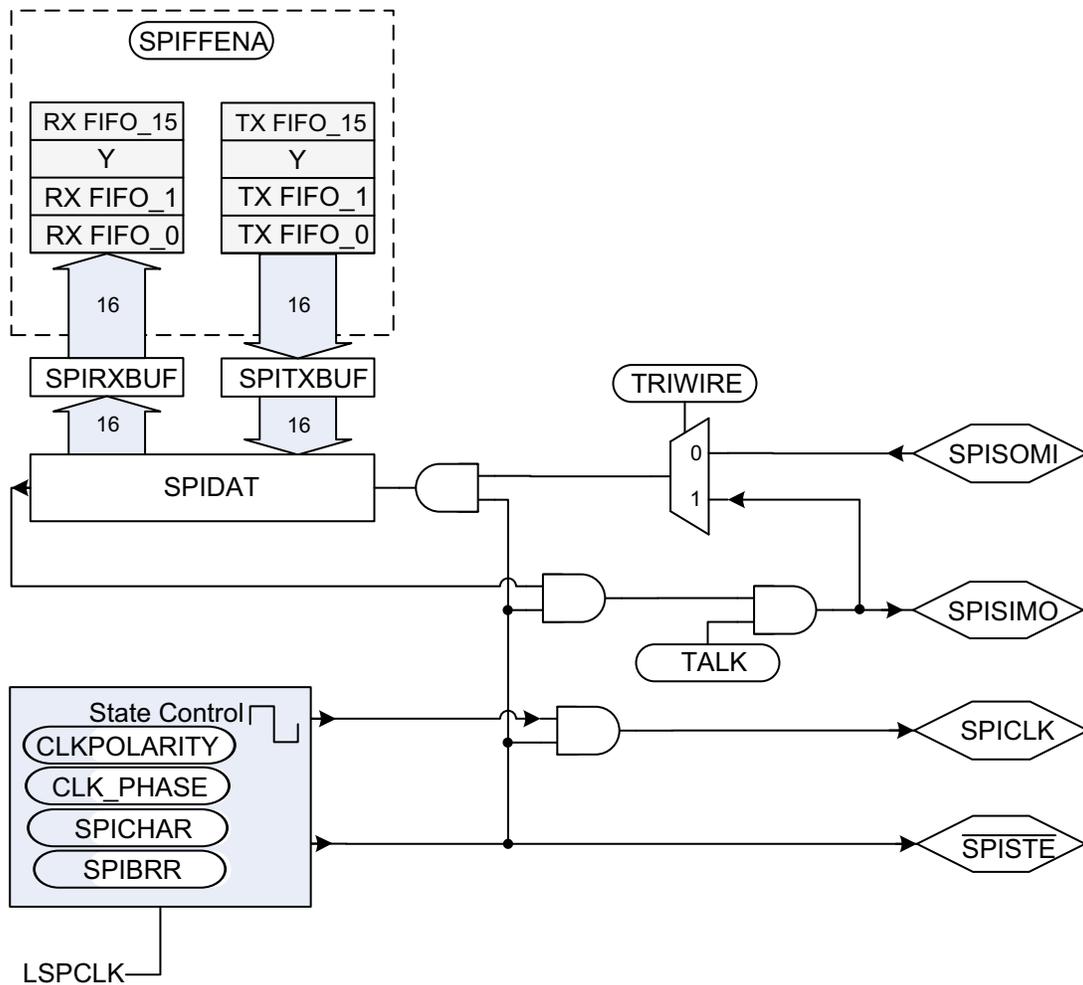


Figure 19-4. SPI Module Master Configuration

### 19.3.3 Slave Mode

In 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 can 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. A character written to the SPITXBUF register is copied to the SPIDAT register when all bits of the current character in SPIDAT have been shifted out. If no character was previously copied to SPIDAT, then any character written to SPITXBUF is immediately copied to SPIDAT. If a character was previously copied to SPIDAT, any data written to SPITXBUF is not copied to SPIDAT until the current character in SPIDAT has been shifted out. 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 SPIDAT has not been previously loaded, the character must be written to SPITXBUF before the beginning of the SPICLK signal.

When the TALK bit is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This makes sure 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 the 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 19-5 is a block diagram of the SPI in slave mode. The block diagram shows the basic control blocks available in SPI slave mode.

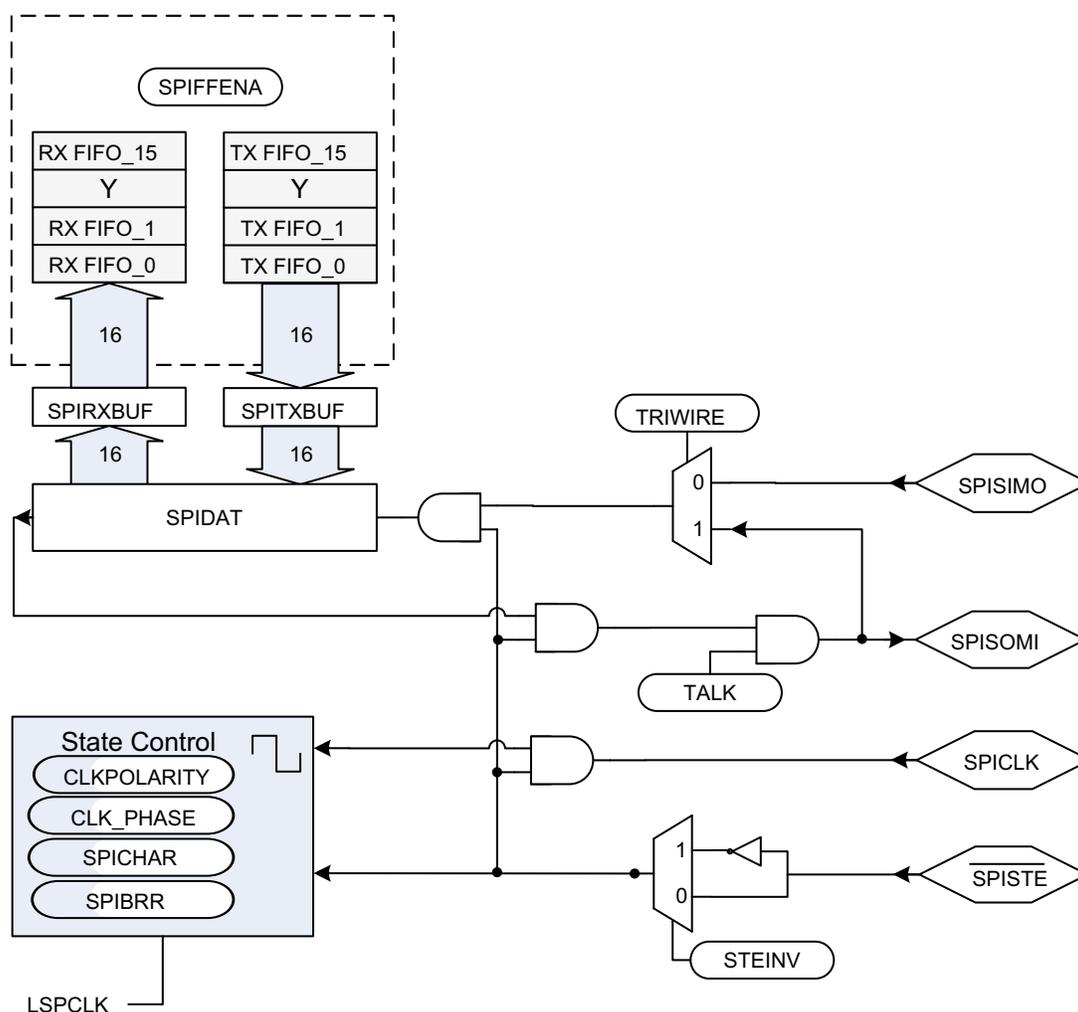


Figure 19-5. SPI Module Slave Configuration

### 19.3.4 Data Format

The four-bit SPICHR register field specifies the number of bits in the data character (1 to 16). This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed.

The following statements apply to characters with fewer than 16 bits:

- Data must be left-justified when written to SPIDAT and SPITXBUF.
- Data read back from SPIRXBUF is right-justified.
- SPIRXBUF contains the most recently received character, right-justified, plus any bits that remain from previous transmission(s) that have been shifted to the left (shown in [Example 19-1](#)).

#### Example 19-1. Transmission of Bit from SPIRXBUF

Conditions:

1. Transmission character length = 1 bit (specified in SPICHR bits)
2. The current value of SPIDAT = 737Bh

SPIDAT (before transmission)																	
	0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	
SPIDAT (after transmission)																	
(TXed) 0 ←	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x <sup>(1)</sup>	← (RXed)
SPIRXBUF (after transmission)																	
	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x <sup>(1)</sup>	

(1) x = 1, if SPISOMI data is high; x = 0, if SPISOMI data is low; master mode is assumed.

### 19.3.5 Baud Rate Selection

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin and can be no greater than the LSPCLK frequency divided by 4.

#### Note

The baud rate must be configured to not exceed the maximum rated GPIO toggle frequency. Refer to the device data sheet for the maximum GPIO toggle frequency.

[Example 19-2](#) shows how to determine the SPI baud rates.

[Example 19-3](#) shows how to calculate the baud rate of the SPI module in standard SPI mode (`HS_MODE = 0`).

#### Example 19-2. Baud Rate Determination

For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$$

For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$$

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 (that is device-specific) and the baud rate at which you are operating.

#### Example 19-3. Baud Rate Calculation in Non-High Speed Mode (`HS_MODE = 0`)

$$\begin{aligned} \text{SPI Baud Rate} &= \frac{\text{LSPCLK}}{\text{SPIBRR} + 1} \quad \text{LSPCLK} = 50 \text{ MHz} \\ &= \frac{50 \times 10^6}{3 + 1} \\ &= 12.5 \text{ Mbps} \end{aligned}$$

### 19.3.6 SPI Clocking Schemes

The clock polarity select bit (CLKPOLARITY) and the clock phase select bit (CLK\_PHASE) control four different clocking schemes on the SPICLK pin. CLKPOLARITY selects the active edge, either rising or falling, of the clock. CLK\_PHASE selects a half-cycle delay of the clock. The four different clocking schemes are:

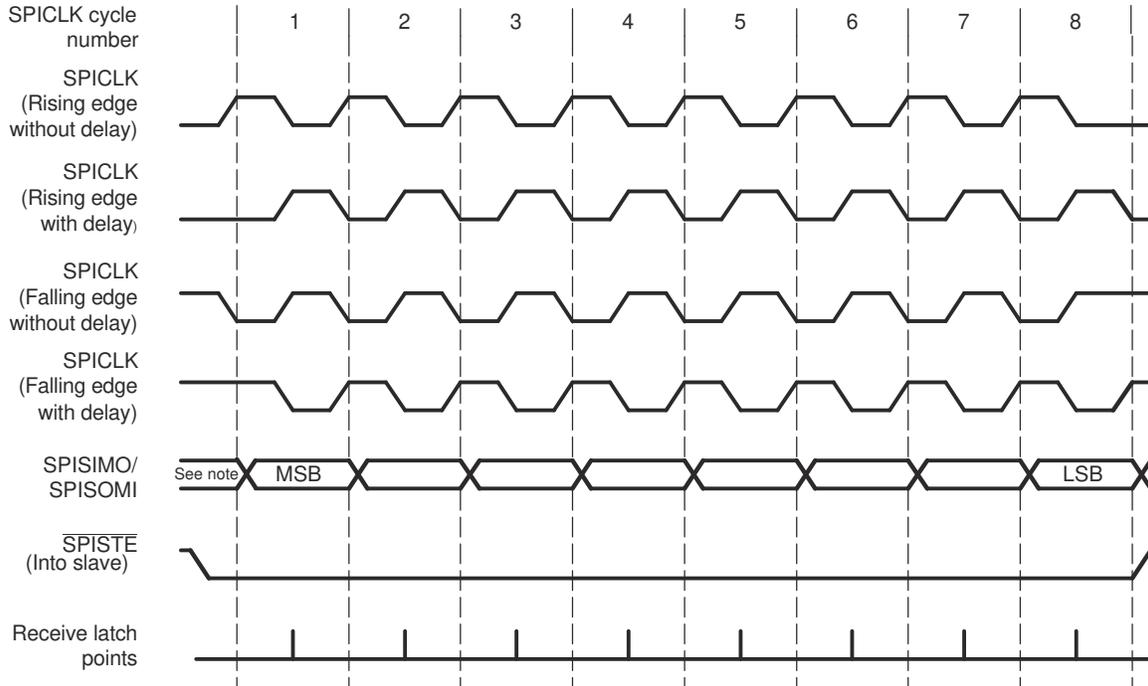
- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK 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 19-3](#). Examples of these four clocking schemes relative to transmitted and received data are shown in [Figure 19-6](#).

**Table 19-3. SPI Clocking Scheme Selection Guide**

SPICLK Scheme	CLKPOLARITY <sup>(1)</sup>	CLK_PHASE <sup>(1)</sup>
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

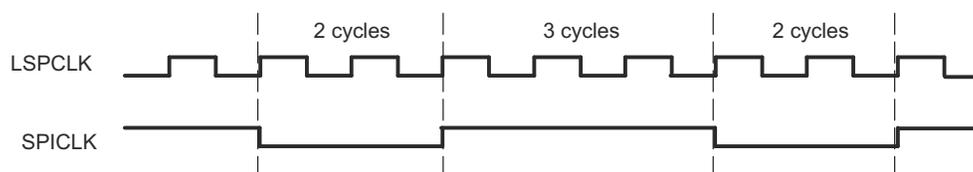
(1) The description of CLK\_PHASE and CLKPOLARITY differs between manufacturers. For proper operation, select the desired waveform to determine the clock phase and clock polarity settings.



**Note:** Previous data bit

**Figure 19-6. SPICLK Signal Options**

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 LSPCLK cycle longer than the high pulse when  $CLKPOLARITY$  bit is clear (0). When  $CLKPOLARITY$  bit is set to 1, the high pulse of the SPICLK is one LSPCLK cycle longer than the low pulse, as shown in Figure 19-7.



**Figure 19-7. SPI: SPICLK-LSPCLK Characteristic when  $(BRR + 1)$  is Odd,  $BRR > 3$ , and  $CLKPOLARITY = 1$**

### 19.3.7 SPI FIFO Description

The following steps explain the FIFO features and help with programming the SPI FIFOs:

1. **Reset.** At reset the SPI powers up in standard SPI mode and the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.
2. **Standard SPI.** The standard 28x SPI mode works with SPIINT/SPIRXINT as the interrupt source.
3. **Mode change.** FIFO mode is enabled by setting the SPIFFENA bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of the operation.
4. **Active registers.** All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT are active.
5. **Interrupts.** FIFO mode has two interrupts: one for the transmit FIFO, SPITXINT; one for the receive FIFO, SPIRXINT. SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI are disabled and this interrupt is serviced as SPI receive FIFO interrupt. For more information, refer to Section 19.2.3.
6. **Buffers.** Transmit and receive buffers are each supplemented with a 16-word FIFO. The one-word transmit buffer (SPITXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer is 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 can define a minimum delay of 0 SPICLK cycles and a maximum of 255 SPICLK cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 255 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 255 SPICLK cycles between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC, and so on.
8. **FIFO status bits.** Both transmit and receive FIFOs have status bits TXFFST or RXFFST that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit (TXFIFO) and receive reset bit (RXFIFO) reset the FIFO pointers to zero when these bits are set to 1. The FIFOs resume operation from start once these bits are cleared to zero.
9. **Programmable interrupt levels.** Both transmit and receive FIFOs can generate CPU interrupts. The transmit interrupt (SPITXINT) is generated whenever the transmit FIFO status bits (TXFFST) match (less than or equal to) the interrupt trigger level bits (TXFFIL). The receive interrupt (SPIRXINT) is generated whenever the receive FIFO status bits (RXFFST) match (greater than or equal to) the interrupt trigger level RXFFIL. This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits is 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

### 19.3.8 SPI High-Speed Mode

The SPI module is capable of reaching full-duplex communication speeds up to LSPCLK/4 (where LSPCLK equals SYSCLK). For the maximum rated speed, refer to the device data sheet.

To achieve the maximum full-duplex speeds, the following restrictions are placed on the design:

- Single master to single slave configuration is supported.
- Loading on the pins must not exceed the value stated in the device data sheet.

When configuring the GPIOs to support high-speed mode, refer to [Section 19.2.2.1](#) for more information.

### 19.3.9 SPI 3-Wire Mode Description

SPI 3-wire mode allows for SPI communication over three pins instead of the normal four pins.

In master mode, if the TRIWIRE bit is set, enabling 3-wire SPI mode, SPISIMOMx 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 SPISIMOMx is no longer used by the SPI.

[Table 19-4](#) indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

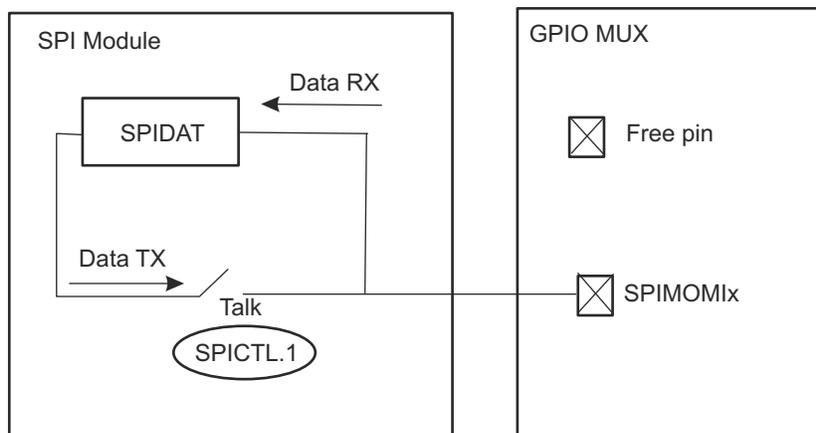
**Table 19-4. 4-wire versus 3-wire SPI Pin Functions**

4-wire SPI	3-wire SPI (Master)	3-wire SPI (Slave)
SPICLKx	SPICLKx	SPICLKx
SPISTEx	SPISTEx	SPISTEx
SPISIMOMx	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. The application software must take care to perform a dummy read to clear the SPI data register of the additional received data.

The TALK bit plays an important role in 3-wire SPI mode. The bit must be set to transmit data and cleared prior to reading data. In master mode, 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 19-8](#) and [Figure 19-9](#) illustrate 3-wire master and slave mode.



**Figure 19-8. SPI 3-wire Master Mode**

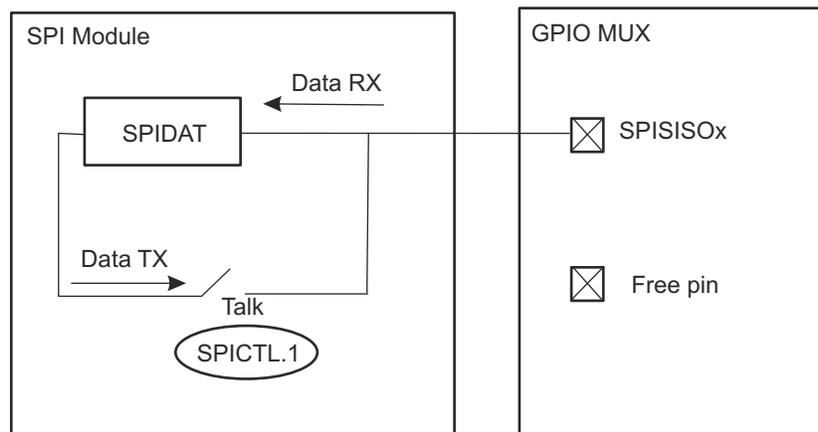

**Figure 19-9. SPI 3-wire Slave Mode**

Table 19-5 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

**Table 19-5. 3-Wire SPI Pin Configuration**

Pin Mode	SPIPRI[TRIWIRE]	SPICTL[TALK]	SPISIMO	SPISOMI
<b>Master Mode</b>				
4-wire	0	X	TX	RX
3-pin mode	1	0	RX	Disconnect from SPI
		1	TX/RX	
<b>Slave Mode</b>				
4-wire	0	X	RX	TX
3-pin mode	1	0	Disconnect from SPI	RX
		1		TX/RX

## 19.4 Programming Procedure

This section describes the procedure for configuring the SPI for the various modes of operation.

### 19.4.1 Initialization Upon Reset

A system reset forces the SPI peripheral into the following default configuration:

- Unit is configured as a slave module (MASTER\_SLAVE = 0)
- Transmit capability is disabled (TALK = 0)
- Data is latched at the input on the falling edge of the SPICLK signal
- Character length is assumed to be one bit
- SPI interrupts are disabled
- Data in SPIDAT is reset to 0000h

### 19.4.2 Configuring the SPI

This section describes the procedure in which to configure the SPI module for operation. To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPISWRESET bit before making initialization changes, and then set this bit after initialization is complete. While the SPI is held in reset (SPISWRESET = 0), configuration can be changed in any order. The following list shows the SPI configuration procedure in a logical order. However, the SPI registers can be written with single 16-bit writes, so the order is not required with the exception of SPISWRESET.

---

#### Note

Do not change the SPI configuration when communication is in progress.

---

To change the SPI configuration:

1. Clear the SPI Software Reset bit (SPISWRESET) to 0 to force the SPI to the reset state.
2. Configure the SPI as desired:
  - Select either master or slave mode (MASTER\_SLAVE).
  - Choose SPICLK polarity and phase (CLKPOLARITY and CLK\_PHASE).
  - Set the desired baud rate (SPIBRR).
  - Enable high-speed mode, if desired (HS\_MODE).
  - Set the SPI character length (SPICHR).
  - Clear the SPI Flags (OVERRUN\_FLAG, INT\_FLAG).
  - Enable  $\overline{\text{SPISTE}}$  inversion (STEINV), if needed.
  - Enable 3-wire mode (TRIWIRE), if needed.
  - If using FIFO enhancements:
    - Enable the FIFO enhancements (SPIFFENA).
    - Clear the FIFO Flags (TXFFINTCLR, RXFFOVFCLR, and RXFFINTCLR).
    - Release transmit and receive FIFO resets (TXFIFO and RXFIFORESET).
    - Release SPI FIFO channels from reset (SPIRST).
3. If interrupts are used:
  - In non-FIFO mode, enable the receiver overrun and/or SPI interrupts (OVERRUNINTENA and SPIINTENA).
  - In FIFO mode, set the transmit and receive interrupt levels (TXFFIL and RXFFIL) then enable the interrupts (TXFFIENA and RXFFIENA).
4. Set SPISWRESET to 1 to release the SPI from the reset state.

### 19.4.3 Configuring the SPI for High-Speed Mode

To achieve the maximum rated speeds, the following settings must be made. This example assumes that the device is operating at 100MHz.

Set LSPCLK equal to SYSCLK:

```
ClkCfgRegs.LOSPCP.bit.LSPCLKDIV = 0;
```

Select the appropriate Pin Mux options in GPIO\_CTRL\_REGS.

During the SPI configuration procedure:

Set HS\_MODE to 1.

```
SpiARegs.SPICCR.bit.HS_MODE = 0x1;
```

Set SPIBRR to 3.  $SPICLK = LSPCLK / (SPIBRR + 1) = 25\text{MHz}$

```
SpiARegs.SPIBRR = 0x3;
```

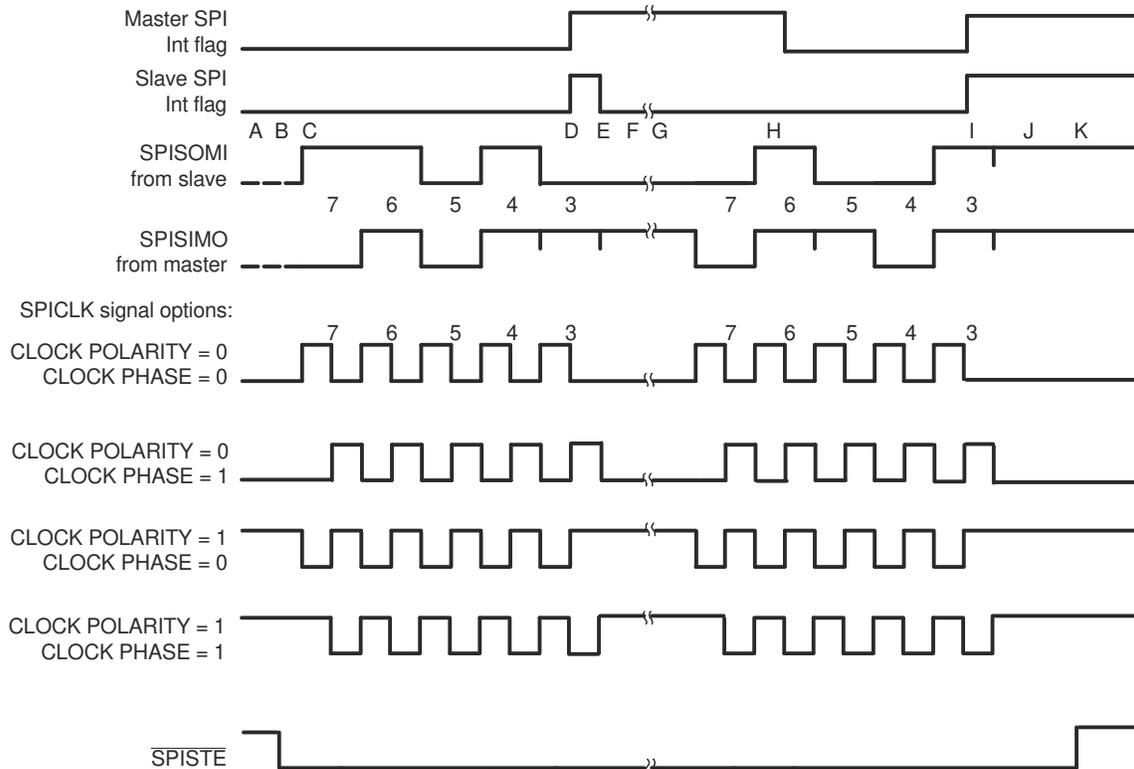
There are no other differences in the configuration from normal SPI operation. Sending and receiving data and interrupts operates without change.

### 19.4.4 Data Transfer Example

The timing diagram shown in Figure 19-10 shows an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK asymmetrical (Figure 19-7) shares similar characterizations with Figure 19-10 except that the data transfer is one LSPCLK cycle longer per bit during the low pulse (CLKPOLARITY = 0) or during the high pulse (CLKPOLARITY = 1) of the SPICLK.

Figure 19-10 is applicable for 8-bit SPI only and is not for C28x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.



- Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
- Master sets the slave  $\overline{\text{SPISTE}}$  signal low (active).
- Master writes 058h to SPIDAT, which starts the transmission procedure.
- First byte is finished and sets the interrupt flags.
- Slave reads 0Bh from the SPIRXBUF (right-justified).
- Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
- Master writes 06Ch to SPIDAT, which starts the transmission procedure.
- Master reads 01Ah from the SPIRXBUF (right-justified).
- Second byte is finished and sets the interrupt flags.
- Master reads 89h and the slave reads 8Dh from the respective SPIRXBUF. After the user software masks off the unused bits, the master receives 09h and the slave receives 0Dh.
- Master clears the slave  $\overline{\text{SPISTE}}$  signal high (inactive).

Figure 19-10. Five Bits per Character

### 19.4.5 SPI 3-Wire Mode Code Examples

In addition to the normal SPI initialization, to configure the SPI module for 3-wire mode, the TRIWIRE bit (SPIPRI.0) must be set to 1. After initialization, there are several considerations to take into account when transmitting and receiving data in 3-wire master and slave mode. The following examples demonstrate these considerations.

In 3-wire master mode, SPICLKx,  $\overline{\text{SPISTEx}}$ , and SPISIMOX pins must be configured as SPI pins (SPISOMIx pin can be configured as non-SPI pin). When the master transmits, the master receives the data the master transmits (because SPISIMOX and SPISOMIx are connected internally in 3-wire mode). Therefore, the junk data received must be cleared from the receive buffer every time data is transmitted.

#### Example 19-4. 3-Wire Master Mode Transmit

```

uint16 data;
uint16 dummy;
SpiRegs.SPICTL.bit.TALK = 1;           // Enable Transmit path
SpiRegs.SPITXBUF = data; // Master transmits data
while(SpiRegs.SPISTS.bit.INT_FLAG !=1) {} // waits until data rx'd
dummy = SpiRegs.SPIRXBUF;             // Clears junk data because
                                       // rx'd same data as tx'd

```

To receive data in 3-wire master mode, the master must clear the TALK (SPICTL.1) bit to 0 to close the transmit path and then transmit dummy data to initiate the transfer from the slave. Because the TALK bit is 0, unlike in transmit mode, the master dummy data does not appear on the SPISIMOX pin, and the master does not receive the dummy data. Instead, the data from the slave is received by the master.

#### Example 19-5. 3-Wire Master Mode Receive

```

uint16 rdata;
uint16 dummy;
SpiRegs.SPICTL.bit.TALK = 0;           // Disable Transmit path
SpiRegs.SPITXBUF = dummy;             // Send dummy to start tx
// NOTE: because TALK = 0, data does not tx onto SPISIMOX pin
while(SpiRegs.SPISTS.bit.INT_FLAG !=1) {} // wait until data received
rdata = SpiRegs.SPIRXBUF;             // Master reads data

```

In 3-wire slave mode, SPICLKx, SPISEx, and SPISOMIx pins must be configured as SPI pins (SPISIMOX pin can be configured as non-SPI pin). Like in master mode, when transmitting, the slave receives the data transmitted and must clear this junk data from the receive buffer.

#### Example 19-6. 3-Wire Slave Mode Transmit

```

uint16 data;
uint16 dummy;
SpiRegs.SPICTL.bit.TALK = 1;           // Enable Transmit path
SpiRegs.SPITXBUF = data;             // Slave transmits data
while(SpiRegs.SPISTS.bit.INT_FLAG !=1) {} // wait until data rx'd
dummy = SpiRegs.SPIRXBUF;             // Clears junk data

```

As in 3-wire master mode, the TALK bit must be cleared to 0. Otherwise, the slave receives data normally.

**Example 19-7. 3-Wire Slave Mode Receive**

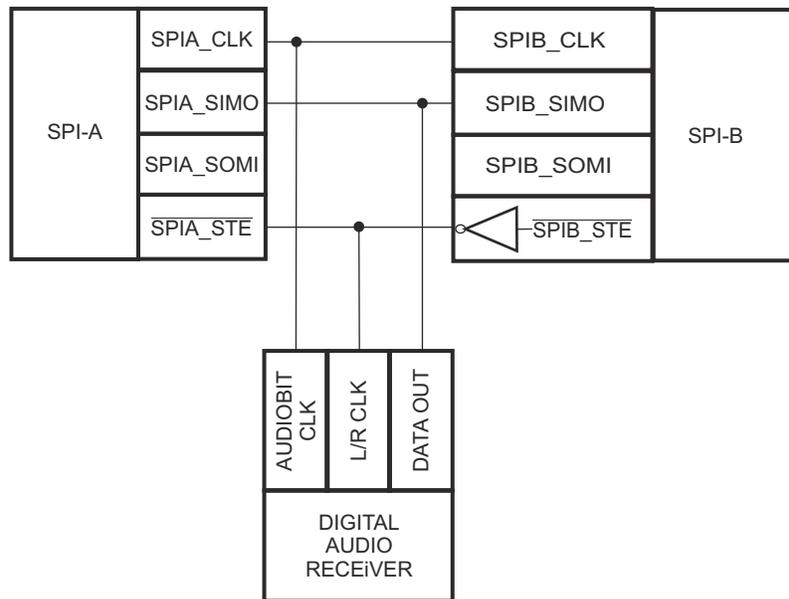
```
Uint16 rdata;
Spiaregs.SPICTL.bit.TALK = 0;
while(SpiaRegs.SPISTS.bit.INT_FLAG !=1) {} // Disable Transmit path
rdata = Spiaregs.SPIRXBUF;                // waits until data rx'd
                                           // Slave reads data
```

### 19.4.6 SPI STEINV Bit in Digital Audio Transfers

On those devices with two SPI modules, enabling the STEINV bit on one of the SPI modules allows the pair of SPIs to receive both left and right-channel digital audio data in slave mode. The SPI module that receives a normal active-low  $\overline{\text{SPISTE}}$  signal stores right-channel data, and the SPI module that receives an inverted active-high  $\overline{\text{SPISTE}}$  signal stores left-channel data from the master. To receive digital audio data from a digital audio interface receiver, the SPI modules can be connected as shown in Figure 19-11.

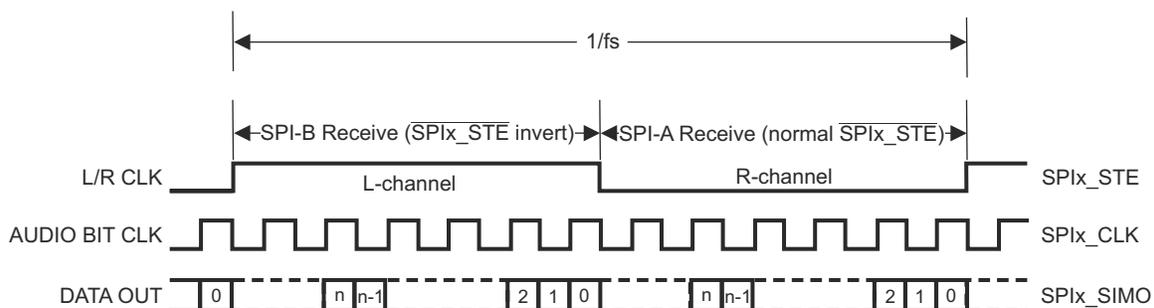
**Note**

This configuration is only applicable to slave mode (MASTER\_SLAVE = 0). When the SPI is configured as master (MASTER\_SLAVE = 1), the STEINV bit has no effect on the  $\overline{\text{SPISTE}}$  pin.



**Figure 19-11. SPI Digital Audio Receiver Configuration Using Two SPIs**

Standard C28x SPI timing requirements limit the number of digital audio interface formats supported using the 2-SPI configuration with the STEINV bit. See the device data sheet electrical specifications for SPI timing requirements. With the SPI clock phase configured such that the CLKPOLARITY bit is 0 and the CLK\_PHASE bit is 1 (data latched on rising edge of clock), standard right-justified digital audio interface data format is supported as shown in Figure 19-12.



**Figure 19-12. Standard Right-Justified Digital Audio Data Format**

## 19.5 Software

### 19.5.1 SPI Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/spi

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 19.5.1.1 SPI Digital Loopback

FILE: spi\_ex1\_loopback.c

This program uses the internal loopback test mode of the SPI module. This is a very basic loopback that does not use the FIFOs or interrupts. A stream of data is sent and then compared to the received stream. The pinmux and SPI modules are configured through the sysconfig file.

The sent data looks like this:

```
0000 0001 0002 0003 0004 0005 0006 0007 .... FFFE FFFF 0000
```

This pattern is repeated forever.

##### *External Connections*

- None

##### *Watch Variables*

- *sData* - Data to send
- *rData* - Received data

#### 19.5.1.2 SPI Digital Loopback with FIFO Interrupts

FILE: spi\_ex2\_loopback\_fifo\_interrupts.c

This program uses the internal loopback test mode of the SPI module. Both the SPI FIFOs and their interrupts are used.

A stream of data is sent and then compared to the received stream. The sent data looks like this:

```
0000 0001
0001 0002
0002 0003
....
FFFE FFFF
FFFF 0000
etc..
```

This pattern is repeated forever.

##### *External Connections*

- None

##### *Watch Variables*

- *sData* - Data to send
- *rData* - Received data
- *rDataPoint* - Used to keep track of the last position in the receive stream for error checking

#### 19.5.1.3 SPI EEPROM

FILE: spi\_ex6\_eeprom.c

This program will write 8 bytes to EEPROM and read them back. The device communicates with the EEPROM via SPI and specific opcodes. This example is written to work with the SPI Serial EEPROM AT25128/256. Note: SPI character length is configured to 8 bits in SysConfig, and not changed throughout the execution of the

program. Runtime updation of character length when CS pin is not controlled by the SPI module can lead to unpredictable behaviour

#### *External Connections*

- Connect external SPI EEPROM
- Connect GPIO08 (PICO) to external EEPROM SI pin
- Connect GPIO10 (POCI) to external EEPROM SO pin
- Connect GPIO09 (CLK) to external EEPROM SCK pin
- Connect GPIO11 (CS) to external EEPROM CS pin
- Connect the external EEPROM VCC and GND pins

#### *Watch Variables*

- writeBuffer - Data that is written to external EEPROM
- readBuffer - Data that is read back from EEPROM
- error - Error count

## 19.6 SPI Registers

This section describes the Serial Peripheral Interface registers. Note that the SPI registers only allow 16-bit accesses.

### 19.6.1 SPI Base Address Table

**Table 19-6. SPI Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
SpiaRegs	SPI_REGS	SPIA_BASE	0x0000_6100	YES

## 19.6.2 SPI\_REGS Registers

Table 19-7 lists the memory-mapped registers for the SPI\_REGS registers. All register offset addresses not listed in Table 19-7 should be considered as reserved locations and the register contents should not be modified.

**Table 19-7. SPI\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	SPICCR	SPI Configuration Control Register		<a href="#">Go</a>
1h	SPICTL	SPI Operation Control Register		<a href="#">Go</a>
2h	SPISTS	SPI Status Register		<a href="#">Go</a>
4h	SPIBRR	SPI Baud Rate Register		<a href="#">Go</a>
6h	SPIRXEMU	SPI Emulation Buffer Register		<a href="#">Go</a>
7h	SPIRXBUF	SPI Serial Input Buffer Register		<a href="#">Go</a>
8h	SPITXBUF	SPI Serial Output Buffer Register		<a href="#">Go</a>
9h	SPIDAT	SPI Serial Data Register		<a href="#">Go</a>
Ah	SPIFFTX	SPI FIFO Transmit Register		<a href="#">Go</a>
Bh	SPIFFRX	SPI FIFO Receive Register		<a href="#">Go</a>
Ch	SPIFFCT	SPI FIFO Control Register		<a href="#">Go</a>
Fh	SPIPRI	SPI Priority Control Register		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 19-8 shows the codes that are used for access types in this section.

**Table 19-8. SPI\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
RC	R C	Read to Clear
Write Type		
W	W	Write
W1C	W 1C	Write 1 to clear
Reset or Default Value		
-n		Value after reset or the default value

### 19.6.2.1 SPICCR Register (Offset = 0h) [Reset = 0000h]

SPICCR is shown in [Figure 19-13](#) and described in [Table 19-9](#).

Return to the [Summary Table](#).

SPICCR controls the setup of the SPI for operation.

**Figure 19-13. SPICCR Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SPISWRESET	CLKPOLARITY	HS_MODE	SPILBK	SPICCHAR			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			

**Table 19-9. SPICCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	SPISWRESET	R/W	0h	<p>SPI Software Reset</p> <p>When changing configuration, you should clear this bit before the changes and set this bit before resuming operation.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. SPISTE will become inactive. SPICLK will be immediately driven to 0 regardless of the clock polarity. The SPI configuration remains unchanged.</p> <p>1h (R/W) = SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register. SPICLK will be returned to its inactive state one SPICLK cycle after this bit is set.</p>
6	CLKPOLARITY	R/W	0h	<p>Shift Clock Polarity</p> <p>This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and POLARITY CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows:</p> <ul style="list-style-type: none"> <li>- CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal. Input data is latched on the falling edge of the SPICLK signal.</li> <li>- CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal. Input data is latched on the rising edge of the SPICLK signal.</li> </ul> <p>1h (R/W) = Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows:</p> <ul style="list-style-type: none"> <li>- CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal. Input data is latched on the rising edge of the SPICLK signal.</li> <li>- CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal. Input data is latched on the falling edge of the SPICLK signal.</li> </ul>

**Table 19-9. 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. Reset type: SYSRSn 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. Reset type: SYSRSn 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. SPICHR = Word length - 1 Reset type: SYSRSn 0h (R/W) = 1-bit word 1h (R/W) = 2-bit word 7h (R/W) = 8-bit word Fh (R/W) = 16-bit word

### 19.6.2.2 SPICTL Register (Offset = 1h) [Reset = 0000h]

SPICTL is shown in [Figure 19-14](#) and described in [Table 19-10](#).

Return to the [Summary Table](#).

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

**Figure 19-14. 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_SLAV E	TALK	SPIINTENA
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 19-10. SPICTL Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	Reserved
4	OVERRUNINTENA	R/W	0h	Overrun Interrupt Enable 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. Reset type: SYSRSn 0h (R/W) = Disable RECEIVER_OVERRUN interrupts. 1h (R/W) = Enable RECEIVER_OVERRUN interrupts.
3	CLK_PHASE	R/W	0h	SPI Clock Phase Select 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. Reset type: SYSRSn 0h (R/W) = Normal SPI clocking scheme, depending on the CLOCK_POLARITY bit (SPICCR.6). 1h (R/W) = SPICLK signal delayed by one half-cycle. Polarity determined by the CLOCK_POLARITY bit.
2	MASTER_SLAVE	R/W	0h	SPI Network Mode Control 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. Reset type: SYSRSn 0h (R/W) = SPI is configured as a slave. 1h (R/W) = SPI is configured as a master.

**Table 19-10. SPICTL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	TALK	R/W	0h	<p><b>Transmit Enable</b>            The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset.</p> <p>Reset type: SYSRSn            0h (R/W) = Disables transmission:            - 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.            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><b>SPI Interrupt Enable</b>            This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit.</p> <p>Reset type: SYSRSn            0h (R/W) = Disables the interrupt.            1h (R/W) = Enables the interrupt.</p>

### 19.6.2.3 SPISTS Register (Offset = 2h) [Reset = 0000h]

SPISTS is shown in [Figure 19-15](#) and described in [Table 19-11](#).

Return to the [Summary Table](#).

SPISTS contains interrupt and status bits.

**Figure 19-15. 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				
W1C-0h	RC-0h	R-0h	R-0h				

**Table 19-11. SPISTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	OVERRUN_FLAG	W1C	0h	<p>SPI Receiver Overrun Flag</p> <p>This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit is cleared in one of three ways:</p> <ul style="list-style-type: none"> <li>- Writing a 1 to this bit</li> <li>- Writing a 0 to SPI SW RESET (SPICCR.7)</li> <li>- Resetting the system</li> </ul> <p>If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow new overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = A receive overrun condition has not occurred.</p> <p>1h (R/W) = The last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application).</p> <p>Writing a '1' will clear this bit. The RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.</p>

**Table 19-11. SPISTS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6	INT_FLAG	RC	0h	<p><b>SPI Interrupt Flag</b>            SPI INT FLAG is a read-only flag. Hardware sets this bit to indicate that the SPI has completed sending or receiving the last bit and is ready to be serviced. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set. The received character is placed in the receiver buffer at the same time this bit is set. This bit is cleared in one of three ways:</p> <ul style="list-style-type: none"> <li>- Reading SPIRXBUF</li> <li>- Writing a 0 to SPI SW RESET (SPICCR.7)</li> <li>- Resetting the system</li> </ul> <p>Note: This bit should not be used if FIFO mode is enabled. The internal process of copying the received word from SPIRXBUF to the Receive FIFO will clear this bit. Use the FIFO status, or FIFO interrupt bits for similar functionality.</p> <p>Reset type: SYSRSn            0h (R/W) = No full words have been received or transmitted.            1h (R/W) = Indicates that the SPI has completed sending or receiving the last bit and is ready to be serviced.</p>
5	BUFFULL_FLAG	R	0h	<p><b>SPI Transmit Buffer Full Flag</b>            This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete.</p> <p>Reset type: SYSRSn            0h (R/W) = Transmit buffer is not full.            1h (R/W) = Transmit buffer is full.</p>
4-0	RESERVED	R	0h	Reserved

### 19.6.2.4 SPIBRR Register (Offset = 4h) [Reset = 0000h]

SPIBRR is shown in [Figure 19-16](#) and described in [Table 19-12](#).

Return to the [Summary Table](#).

SPIBRR contains the bits used for baud-rate selection.

**Figure 19-16. SPIBRR Register**

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

**Table 19-12. SPIBRR Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6-0	SPI_BIT_RATE	R/W	0h	<p>SPI Baud Rate Control</p> <p>These bits determine the bit transfer rate if the SPI is the network SPI BIT RATE 0 master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.)</p> <p>If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master. Therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's LSPCLK signal divided by 4.</p> <p>In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula:</p> <p>For SPIBRR = 3 to 127: SPI Baud Rate = LSPCLK / (SPIBRR + 1)</p> <p>For SPIBRR = 0, 1, or 2: SPI Baud Rate = LSPCLK / 4</p> <p>Reset type: SYSRSn</p> <p>3h (R/W) = SPI Baud Rate = LSPCLK/4</p> <p>4h (R/W) = SPI Baud Rate = LSPCLK/5</p> <p>7Eh (R/W) = SPI Baud Rate = LSPCLK/127</p> <p>7Fh (R/W) = SPI Baud Rate = LSPCLK/128</p>

### 19.6.2.5 SPIRXEMU Register (Offset = 6h) [Reset = 0000h]

SPIRXEMU is shown in [Figure 19-17](#) and described in [Table 19-13](#).

Return to the [Summary Table](#).

SPIRXEMU contains the received data. Reading SPIRXEMU does not clear the SPI INT FLAG bit of SPISTS. This is not a real register but a dummy address from which the contents of SPIRXBUF can be read by the emulator without clearing the SPI INT FLAG.

**Figure 19-17. SPIRXEMU Register**

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

**Table 19-13. SPIRXEMU Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	ERXBn	R	0h	<p>Emulation Buffer Received Data</p> <p>SPIRXEMU functions almost identically to SPIRXBUF, except that reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). Once the SPIDAT has received the complete character, the character is transferred to SPIRXEMU and SPIRXBUF, where it can be read. At the same time, SPI INT FLAG is set.</p> <p>This mirror register was created to support emulation. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. SPIRXEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIRXEMU does not clear the SPI INT FLAG bit, but reading SPIRXBUF clears this flag. In other words, SPIRXEMU enables the emulator to emulate the true operation of the SPI more accurately.</p> <p>It is recommended that you view SPIRXEMU in the normal emulator run mode.</p> <p>Reset type: SYSRSn</p>

### 19.6.2.6 SPIRXBUF Register (Offset = 7h) [Reset = 0000h]

SPIRXBUF is shown in [Figure 19-18](#) and described in [Table 19-14](#).

Return to the [Summary Table](#).

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit in SPISTS. If FIFO mode is enabled, reading this register will also decrement the RXFFST counter in SPIFFRX.

**Figure 19-18. SPIRXBUF Register**

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

**Table 19-14. SPIRXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	RXBn	R	0h	Received Data 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. Reset type: SYSRSn

### 19.6.2.7 SPITXBUF Register (Offset = 8h) [Reset = 0000h]

SPITXBUF is shown in [Figure 19-19](#) and described in [Table 19-15](#).

Return to the [Summary Table](#).

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit in SPISTS. When the transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set. In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

**Figure 19-19. SPITXBUF Register**

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

**Table 19-15. SPITXBUF Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	TXBn	R/W	0h	Transmit Data Buffer This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. Writes to SPITXBUF must be left-justified. Reset type: SYSRSn

### 19.6.2.8 SPIDAT Register (Offset = 9h) [Reset = 0000h]

SPIDAT is shown in [Figure 19-20](#) and described in [Table 19-16](#).

Return to the [Summary Table](#).

SPIDAT is the transmit and receive shift register. Data written to SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit (MSB) shifted out of the SPI, a bit is shifted into the LSB end of the shift register.

**Figure 19-20. SPIDAT Register**

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

**Table 19-16. SPIDAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-0	SDATn	R/W	0h	Serial Data Shift Register - 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, check the CLOCK POLARITY bit (SPICCR.6) described in Section 10.2.1.1 and the CLOCK PHASE bit (SPICTL.3) described in Section 10.2.1.2, for the requirements. 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. Reset type: SYSRSn

### 19.6.2.9 SPIFFTX Register (Offset = Ah) [Reset = A000h]

SPIFFTX is shown in [Figure 19-21](#) and described in [Table 19-17](#).

Return to the [Summary Table](#).

SPIFFTX contains both control and status bits related to the output FIFO buffer. This includes FIFO reset control, FIFO interrupt level control, FIFO level status, as well as FIFO interrupt enable and clear bits.

**Figure 19-21. SPIFFTX Register**

15	14	13	12	11	10	9	8
SPIRST	SPIFFENA	TXFIFO				TXFFST	
R/W-1h	R/W-0h	R/W-1h				R-0h	
7	6	5	4	3	2	1	0
TXFFINT	TXFFINTCLR	TXFFIENA				TXFFIL	
R-0h	W-0h	R/W-0h				R/W-0h	

**Table 19-17. SPIFFTX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	SPIRST	R/W	1h	SPI Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is. 1h (R/W) = SPI FIFO can resume transmit or receive. No effect to the SPI registers bits.
14	SPIFFENA	R/W	0h	SPI FIFO Enhancements Enable Reset type: SYSRSn 0h (R/W) = SPI FIFO enhancements are disabled. 1h (R/W) = SPI FIFO enhancements are enabled.
13	TXFIFO	R/W	1h	TX FIFO Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Release transmit FIFO from reset.
12-8	TXFFST	R	0h	Transmit FIFO Status Reset type: SYSRSn 0h (R/W) = Transmit FIFO is empty. 1h (R/W) = Transmit FIFO has 1 word. 2h (R/W) = Transmit FIFO has 2 words. 10h (R/W) = Transmit FIFO has 16 words, which is the maximum. 1Fh (R/W) = Reserved.
7	TXFFINT	R	0h	TX FIFO Interrupt Flag Reset type: SYSRSn 0h (R/W) = TXFIFO interrupt has not occurred, This is a read-only bit. 1h (R/W) = TXFIFO interrupt has occurred, This is a read-only bit.
6	TXFFINTCLR	W	0h	TXFIFO Interrupt Clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on 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 Reset type: SYSRSn 0h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be disabled. 1h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) will be enabled.

**Table 19-17. SPIFFTX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	TXFFIL	R/W	0h	Transmit FIFO Interrupt Level Bits Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4-0) and FIFO level bits (TXFFIL4-0) match (less than or equal to). Reset type: SYSRSn 0h (R/W) = A TX FIFO interrupt request is generated when there are no words remaining in the TX buffer. 1h (R/W) = A TX FIFO interrupt request is generated when there is 1 word or no words remaining in the TX buffer. 2h (R/W) = A TX FIFO interrupt request is generated when there is 2 words or fewer remaining in the TX buffer. 10h (R/W) = A TX FIFO interrupt request is generated when there are 16 words or fewer remaining in the TX buffer. 1Fh (R/W) = Reserved.

### 19.6.2.10 SPIFFRX Register (Offset = Bh) [Reset = 201Fh]

SPIFFRX is shown in [Figure 19-22](#) and described in [Table 19-18](#).

Return to the [Summary Table](#).

SPIFFRX contains both control and status bits related to the input FIFO buffer. This includes FIFO reset control, FIFO interrupt level control, FIFO level status, as well as FIFO interrupt enable and clear bits.

**Figure 19-22. SPIFFRX Register**

15	14	13	12	11	10	9	8	
RXFFOVF	RXFFOVFCLR	RXFIFORESET	RXFFST					
R-0h	W-0h	R/W-1h	R-0h					
7	6	5	4	3	2	1	0	
RXFFINT	RXFFINTCLR	RXFFIENA	RXFFIL					
R-0h	W-0h	R/W-0h	R/W-1Fh					

**Table 19-18. SPIFFRX Register Field Descriptions**

Bit	Field	Type	Reset	Description
15	RXFFOVF	R	0h	Receive FIFO Overflow Flag Reset type: SYSRSn 0h (R/W) = Receive FIFO has not overflowed. This is a read-only bit. 1h (R/W) = Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVFCLR	W	0h	Receive FIFO Overflow Clear Reset type: SYSRSn 0h (R/W) = Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero. 1h (R/W) = Write 1 to clear SPIFFRX[RXFFOVF].
13	RXFIFORESET	R/W	1h	Receive FIFO Reset Reset type: SYSRSn 0h (R/W) = Write 0 to reset the FIFO pointer to zero, and hold in reset. 1h (R/W) = Re-enable receive FIFO operation.
12-8	RXFFST	R	0h	Receive FIFO Status Reset type: SYSRSn 0h (R/W) = Receive FIFO is empty. 1h (R/W) = Receive FIFO has 1 word. 2h (R/W) = Receive FIFO has 2 words. 10h (R/W) = Receive FIFO has 16 words, which is the maximum. 1Fh (R/W) = Reserved.
7	RXFFINT	R	0h	Receive FIFO Interrupt Flag Reset type: SYSRSn 0h (R/W) = RXFIFO interrupt has not occurred. This is a read-only bit. 1h (R/W) = RXFIFO interrupt has occurred. This is a read-only bit.
6	RXFFINTCLR	W	0h	Receive FIFO Interrupt Clear Reset type: SYSRSn 0h (R/W) = Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero. 1h (R/W) = Write 1 to clear SPIFFRX[RXFFINT] flag
5	RXFFIENA	R/W	0h	RX FIFO Interrupt Enable Reset type: SYSRSn 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 19-18. SPIFFRX Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	RXFFIL	R/W	1Fh	<p>Receive FIFO Interrupt Level Bits</p> <p>Receive FIFO generates an interrupt when the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The default value of these bits after reset is 11111. This avoids frequent interrupts after reset, as the receive FIFO will be empty most of the time.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = A RX FIFO interrupt request is generated when there is 0 or more words in the RX buffer.</p> <p>1h (R/W) = A RX FIFO interrupt request is generated when there are 1 or more words in the RX buffer.</p> <p>2h (R/W) = A RX FIFO interrupt request is generated when there are 2 or more words in the RX buffer.</p> <p>10h (R/W) = A RX FIFO interrupt request is generated when there are 16 words in the RX buffer.</p> <p>1Fh (R/W) = Reserved.</p>

### 19.6.2.11 SPIFFCT Register (Offset = Ch) [Reset = 0000h]

SPIFFCT is shown in [Figure 19-23](#) and described in [Table 19-19](#).

Return to the [Summary Table](#).

SPIFFCT controls the FIFO transmit delay bits.

**Figure 19-23. SPIFFCT Register**

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

**Table 19-19. SPIFFCT Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	TXDLY	R/W	0h	<p>FIFO Transmit Delay Bits</p> <p>These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 255 serial clock cycles. In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In the FIFO mode TXBUF should not be treated as one additional level of buffer.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF immediately upon completion of transmission of the previous word.</p> <p>1h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF1 serial clock cycle after completion of transmission of the previous word.</p> <p>2h (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF 2 serial clock cycles after completion of transmission of the previous word.</p> <p>FFh (R/W) = The next word in the TX FIFO buffer is transferred to SPITXBUF 255 serial clock cycles after completion of transmission of the previous word.</p>

### 19.6.2.12 SPIPRI Register (Offset = Fh) [Reset = 0000h]

SPIPRI is shown in [Figure 19-24](#) and described in [Table 19-20](#).

Return to the [Summary Table](#).

SPIPRI controls auxiliary functions for the SPI including emulation control, SPISTE inversion, and 3-wire control.

**Figure 19-24. SPIPRI Register**

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	RESERVED	SOFT	FREE	RESERVED		STEINV	TRIWIRES
R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h		R/W-0h	R/W-0h

**Table 19-20. SPIPRI Register Field Descriptions**

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	RESERVED	R/W	0h	Reserved
5	SOFT	R/W	0h	<p>Emulation Soft Run</p> <p>This bit only has an effect when the FREE bit is 0.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Transmission stops midway in the bit stream while TSUSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF are shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication freezes right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI starts transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point.</p> <p>1h (R/W) = If the emulation suspend occurs before the start of a transmission, (that is, before the first SPICLK pulse) then the transmission will not occur. If the emulation suspend occurs after the start of a transmission, then the data will be shifted out to completion. When the start of transmission occurs is dependent on the baud rate used.</p> <p>Standard SPI mode: Stop after transmitting the words in the shift register and buffer. That is, after TXBUF and SPIDAT are empty.</p> <p>In FIFO mode: Stop after transmitting the words in the shift register and buffer. That is, after TX FIFO and SPIDAT are empty.</p>
4	FREE	R/W	0h	<p>Emulation Free Run</p> <p>These bits determine what occurs when an emulation suspend occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode) or, if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = Emulation mode is selected by the SOFT bit</p> <p>1h (R/W) = Free run, continue SPI operation regardless of suspend or when the suspend occurred.</p>
3-2	RESERVED	R	0h	Reserved
1	STEINV	R/W	0h	<p>SPISTEn Inversion Bit</p> <p>On devices with 2 SPI modules, inverting the SPISTE signal on one of the modules allows the device to receive left and right- channel digital audio data.</p> <p>This bit is only applicable to slave mode. Writing to this bit while configured as master (MASTER_SLAVE = 1) has no effect</p> <p>Reset type: SYSRSn</p> <p>0h (R/W) = SPISTEn is active low (normal)</p> <p>1h (R/W) = SPISTE is active high (inverted)</p>

**Table 19-20. SPIPRI Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	TRIWIRE	R/W	0h	SPI 3-wire Mode Enable Reset type: SYSRSn 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.

### 19.6.3 SPI Registers to Driverlib Functions

**Table 19-21. SPI Registers to Driverlib Functions**

File	Driverlib Function
<b>SPICCR</b>	
spi.c	SPI_setConfig
spi.c	SPI_clearInterruptStatus
spi.c	SPI_pollingNonFIFOTransaction
spi.c	SPI_pollingFIFOTransaction
spi.h	SPI_enableModule
spi.h	SPI_disableModule
spi.h	SPI_setcharLength
spi.h	SPI_enableLoopback
spi.h	SPI_disableLoopback
spi.h	SPI_enableHighSpeedMode
spi.h	SPI_disableHighSpeedMode
<b>SPICTL</b>	
spi.c	SPI_setConfig
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.h	SPI_enableTalk
spi.h	SPI_disableTalk
<b>SPISTS</b>	
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_writeDataBlockingNonFIFO
spi.h	SPI_readDataBlockingNonFIFO
<b>SPIBRR</b>	
spi.c	SPI_setConfig
spi.c	SPI_setBaudRate
<b>SPIRXEMU</b>	
spi.h	SPI_readRxEmulationBuffer
<b>SPIRXBUF</b>	
spi.h	SPI_readDataNonBlocking
spi.h	SPI_readDataBlockingFIFO
spi.h	SPI_readDataBlockingNonFIFO
<b>SPITXBUF</b>	
spi.h	SPI_writeDataNonBlocking
spi.h	SPI_writeDataBlockingFIFO

**Table 19-21. SPI Registers to Driverlib Functions (continued)**

File	Driverlib Function
spi.h	SPI_writeDataBlockingNonFIFO
<b>SPIDAT</b>	
-	
<b>SPIFFTX</b>	
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_enableFIFO
spi.h	SPI_disableFIFO
spi.h	SPI_resetTxFIFO
spi.h	SPI_setFIFOInterruptLevel
spi.h	SPI_getFIFOInterruptLevel
spi.h	SPI_getTxFIFOStatus
spi.h	SPI_isBusy
spi.h	SPI_reset
<b>SPIFFRX</b>	
spi.c	SPI_enableInterrupt
spi.c	SPI_disableInterrupt
spi.c	SPI_getInterruptStatus
spi.c	SPI_clearInterruptStatus
spi.h	SPI_enableFIFO
spi.h	SPI_disableFIFO
spi.h	SPI_resetRxFIFO
spi.h	SPI_setFIFOInterruptLevel
spi.h	SPI_getFIFOInterruptLevel
spi.h	SPI_getRxFIFOStatus
<b>SPIFFCT</b>	
spi.h	SPI_setTxFifoTransmitDelay
<b>SPIPRI</b>	
spi.h	SPI_enableTriWire
spi.h	SPI_disableTriWire
spi.h	SPI_setPTESignalPolarity
spi.h	SPI_setEmulationMode

Chapter 20  
**Embedded Pattern Generator (EPG)**

---



This chapter describes the Embedded Pattern Generator (EPG) module.

<b>20.1 Introduction</b> .....	<b>1876</b>
<b>20.2 Clock Generator Modules</b> .....	<b>1878</b>
<b>20.3 Signal Generator Module</b> .....	<b>1880</b>
<b>20.4 EPG Peripheral Signal Mux Selection</b> .....	<b>1883</b>
<b>20.5 EPG Example Use Cases</b> .....	<b>1885</b>
<b>20.6 EPG Interrupt</b> .....	<b>1888</b>
<b>20.7 Software</b> .....	<b>1889</b>
<b>20.8 EPG Registers</b> .....	<b>1890</b>

## 20.1 Introduction

The Embedded Pattern Generator (EPG) module is a customizable pattern and clock generator that can serve many test and application scenarios that require a simple pattern generator or a periodic clock generator. The EPG module can also be used to capture an incoming serial stream of data.

### 20.1.1 Features

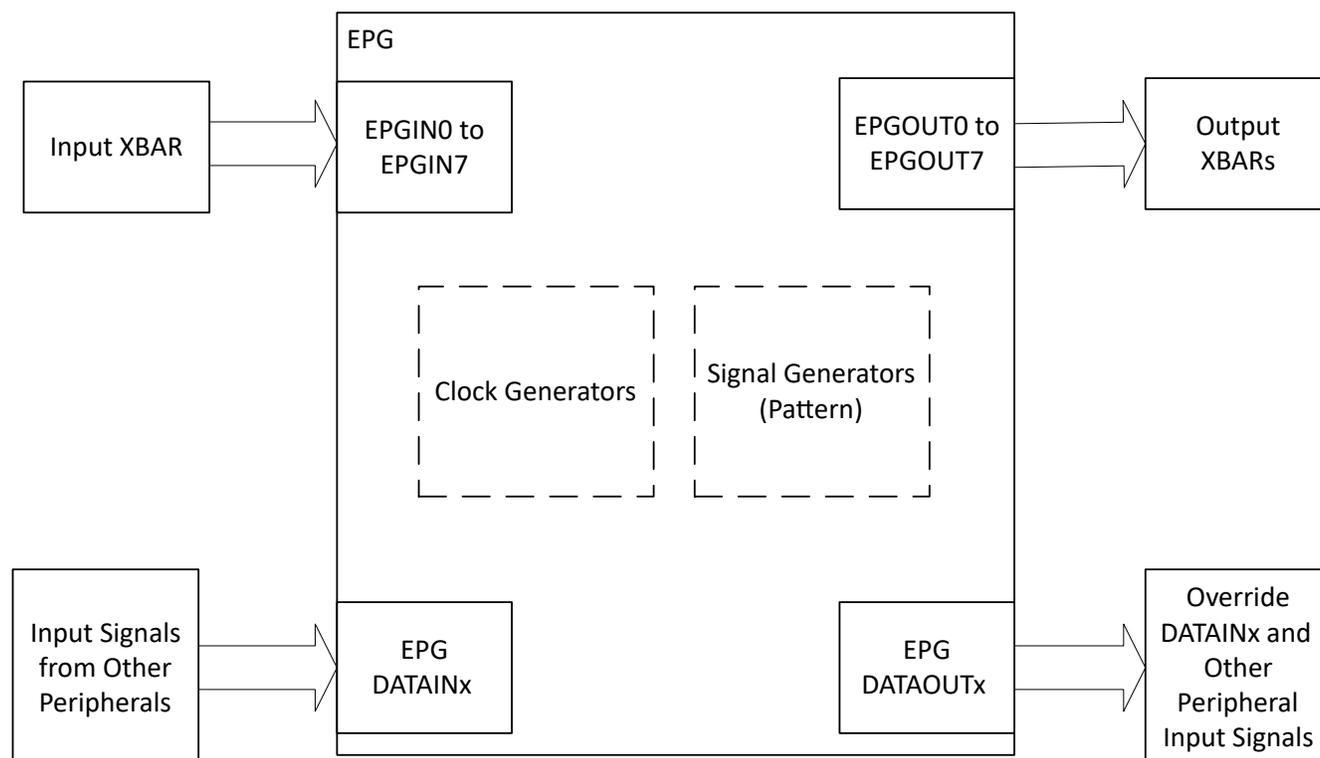
Features of the EPG module are:

- Clock generation:
  - Independent clock generation and clock division
  - Synchronous clock generation with programmable offsets
- Pattern generation:
  - Independent serial data stream generation
  - Serial data stream and the associated clock generation
  - Ability to skew clock with respect to serial data
  - Synchronous data stream with programmable offset with respect to one another

The EPG output signals are connected to GPIOs or other peripherals inside the device. The EPG inputs act as shift register inputs to capture incoming serial data streams. This allows the EPG to be configured as a custom serial module. The EPG is also capable of generating interrupts that are used to supply the pattern generators with new data or signal the completion of a generated pattern.

### 20.1.2 EPG Block Diagram

Figure 20-1 shows the EPG overview block diagram.



**Figure 20-1. EPG Overview Block Diagram**

Figure 20-2 shows the EPG block diagram.

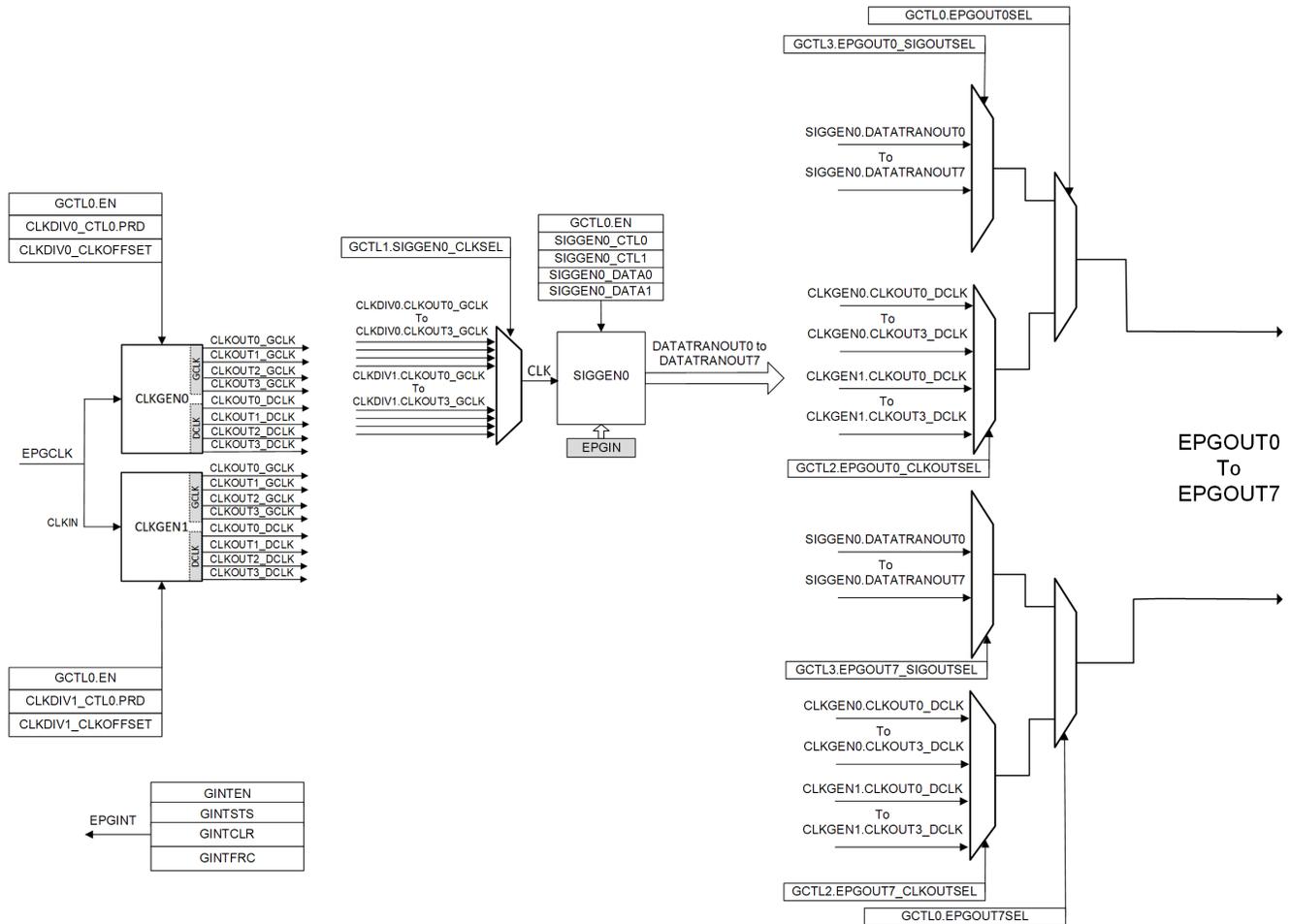


Figure 20-2. EPG Detailed Block Diagram

**Note**

The number of available EPGx SIGGEN modules for each device can be different. This diagram shows an EPG with 1 SIGGEN module (SIGGEN0). Refer to the EPG\_REGS register description to see how many SIGGEN modules are available for a specific device.

The EPG input clock (EPGCLK) is sourced from PERx.SYSCLK.

**20.1.3 EPG Related Collateral**

**Foundational Materials**

- [C2000 Academy - EPG](#)
- [C2000 Embedded Pattern Generator \(Video\)](#)

**Getting Started Materials**

- [Designing With the C2000™ Embedded Pattern Generator \(EPG\) Application Report](#)

## 20.2 Clock Generator Modules

The clock generator modules use the EPG input clock and generate four output clocks, see [Figure 20-3](#). Each of the four output clocks (CLKOUT0 to CLKOUT3) has a DCLK and a GCLK version. The EPG clock division results in a gated, GCLK, version of the EPG input clock which are named CLKOUT0\_GCLK to CLKOUT3\_GCLK. The other version, DCLK (CLKOUT0\_DCLK to CLKOUT3\_DCLK), have the same period but with an approximately 50% duty cycle. The clock generation takes place using the divider settings CLKDIVx\_CTL0.PRD, and clock offset settings CLKDIVx\_CLK.CLK<sub>y</sub>OFFSET. The RUNCLOCK signal generated by the clock stop logic determines when the clock generation is started and stopped.

The clock divider counter is a simple up counter, which has a period determined by CLKDIVx\_CTL0.PRD. This divider counter begins counting when RUNCLOCK is set. The CLKOUT0 to CLKOUT4 GCLKs are gated versions of the CLKIN (EPG input clock). The clock gates are enabled when the counter value matches the corresponding clock offsets. In the case where RUNCLOCK is cleared, the clock gates are disabled and the counter is set to zero.

### Note

The CLKDIVx\_CTL0.PRD register can only be written when GCTL0.EN is 0.

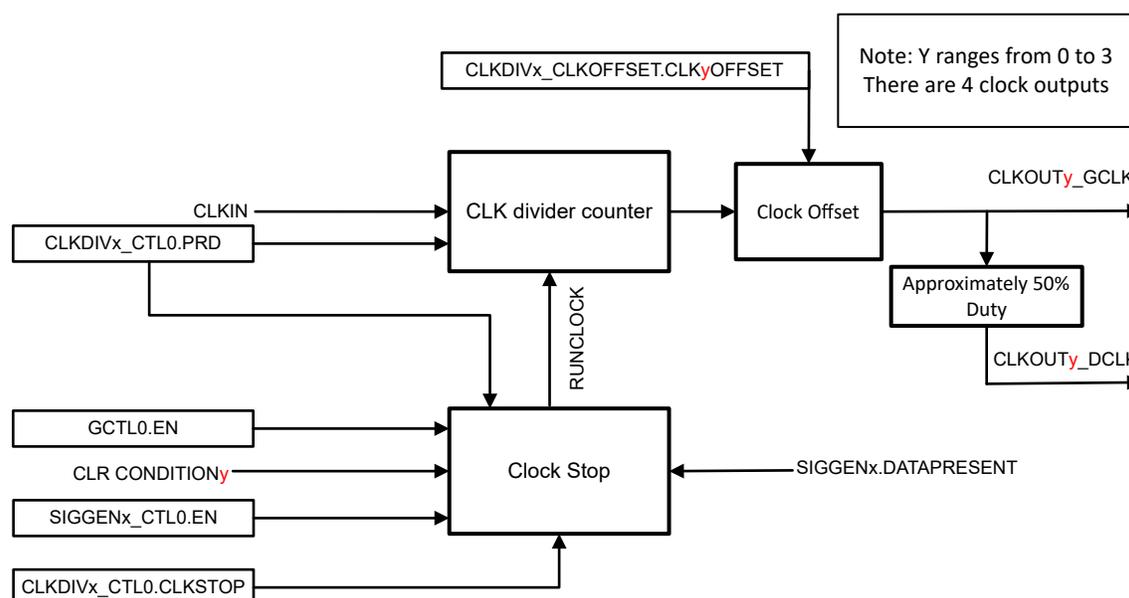


Figure 20-3. EPG Clock Generator

### 20.2.1 DCLK (50% duty cycle clock)

The CLKOUT<sub>y</sub>\_DCLK is generated by setting the clock output for half the clock divider period. When CLKDIVx\_CTL0.PRD is set to zero, CLKOUT<sub>y</sub>\_DCLK is the same as the input clock.

## 20.2.2 Clock Stop

Figure 20-4 shows how the Clock Stop module sets and clears the RUNCLOCK signal.

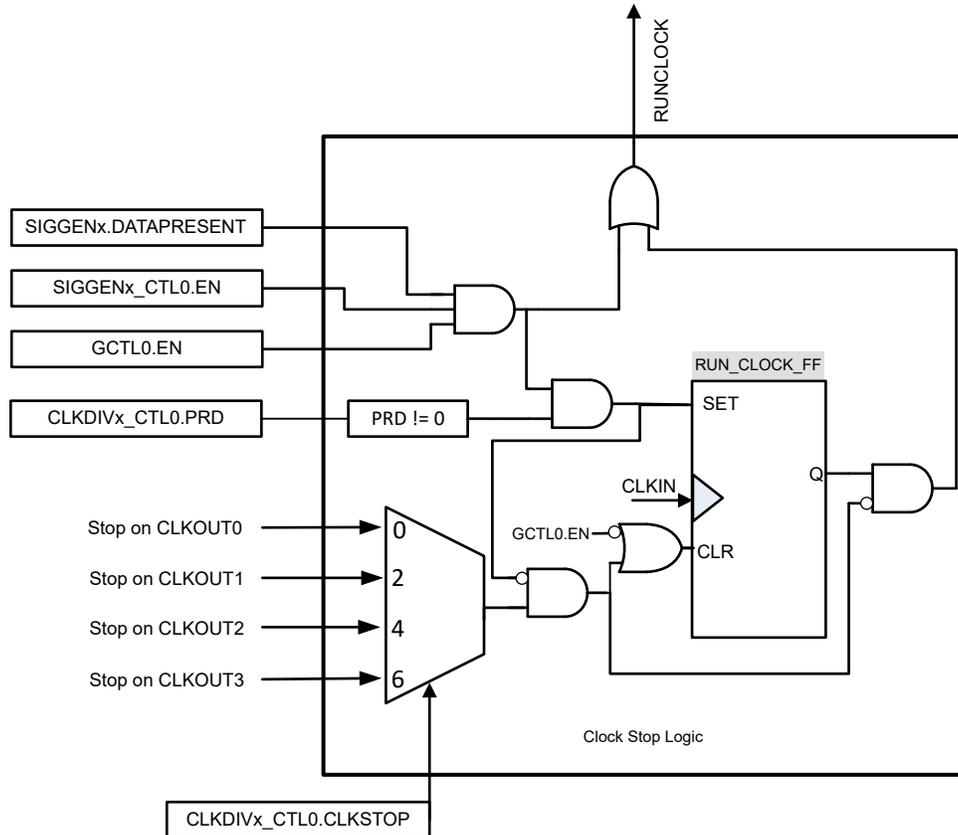


Figure 20-4. EPG Clock Stop

In signal generator modes, SIGGENx\_CTL0.EN bit is cleared when BIT\_LENGTH shifts (or rotates) is completed. This makes sure that data is not shifted out. In addition, clock generation (generation of CLKOUT0 to CLKOUT3) also is stopped to make sure that sampling of input data does not continue after BIT\_LENGTH shifts when SIGGENx\_CTL0.EN is cleared to 0.

The RUNCLOCK signal has to be high for the clock generation circuitry to be active. When SIGGENx\_CTL0.EN is cleared, the clock generation can be selected to stop on the falling edge of CLKOUT0 to CLKOUT3.

The clock stop module operates as follow:

- RUNCLOCK is asserted as soon as both SIGGENx\_CTL0.EN and GCTL0.EN are set.
- RUN\_CLOCK\_FF is set when both SIGGENx\_CTL0.EN and GCTL0.EN are set and CLKDIVx\_CTL0.PRD is not zero.
- When SIGGENx\_CTL0.EN is cleared, RUN\_CLOCK\_FF is not cleared immediately. This makes sure that RUNCLOCK remains set and the clock generation circuitry remains enabled.
- When SIGGENx.DATA0 and SIGGENx.DATA1 are not filled with data before the next shift sequence (repeat shift modes) and signal generator is waiting for data to be written, clocks are stopped.
- The RUN\_CLOCK\_FF can be cleared on the falling edge of CLKOUT0 to CLKOUT3 based on the configuration of CLKSTOP field.
- Clearing GCTL0.EN clears RUN\_CLOCK\_FF unconditionally.

## 20.3 Signal Generator Module

The signal generator module is the main component of the EPG and generates the data stream that follows custom patterns. Figure 20-5 shows the main components. This module has 8 output ports DATATRANOUT0 to DATATRANOUT7. The two registers SIGGENx\_DATA1 and SIGGENx\_DATA0 constitute a 64-bit bus named DATA[63:0]. DATATRANIN[63:0], which is used for all the data transform operations, can be DATA[63:0] or bit reversed DATA (when SIGGENx\_CTL0.BRIN bit is set). The DATATRANOUT0 to DATATRANOUT7 are connected to DATATRANIN0 to DATATRANIN7 when the signal generator is not in BIT\_BANG mode. In BIT\_BANG mode, DATATRANOUT0 to DATATRANOUT7 are connected to DATATRANIN0, DATATRANIN8, and DATATRANIN16 to DATATRANIN56.

In addition to generating data outputs, one of the 8 EPGIN inputs can act as data input to SIGGENx\_DATAy registers. In Figure 20-5, the EPGIN\_MUX block illustrates the mechanism used to capture the input data stream. This enables one to capture a data input stream using EPG module.

Data transformation is done on the DATATRANIN bus, and is determined by the configured mode (SIGGENx\_CTL0.MODE), provided GCTL0.EN and SIGGENx\_CTL0.EN are both set. If either of these enable bits are 0, then the data output of the transform block is the same as the input. The transformed output is bit reversed when SIGGENx\_CTL0.BROUT bit is set. Conditions under which the DATA active register (SIGGENx\_DATA0\_ACTIVE and SIGGENx\_DATA1\_ACTIVE) are:

- Loaded from SIGGENx\_DATA1 and SIGGENx\_DATA0
- Directly updated on a memory mapped write to SIGGENx\_DATA1 and SIGGENx\_DATA0
- Holds the current data

**Table 20-1. SIGGENx Active Register Loading**

Condition	SIGGENx_DATA1_ACTIVE DATA ACTIVE Register [63:32]	SIGGENx_DATA0_ACTIVE DATA ACTIVE Register [31:0]
Memory mapped write to SIGGENx_DATA0 register and SIGGENx_CTL0.EN is 0	No updates	Updated with the value written to SIGGENx_DATA0 register
Memory mapped write to SIGGENx_DATA1 register and SIGGENx_CTL0.EN is 0.	Updated with the value written to SIGGENx_DATA1 register	No updates
“BITLENGTH” number of shifts are done, and Mode is SHIFT_RIGHT_REPEAT or SHIFT_LEFT_REPEAT, and BITLENGTH <= 32, and Either SIGGENx_DATA0 or SIGGENx_DATA1 has been updated	Copy SIGGENx_DATA1 register content	Copy SIGGENx_DATA0 register content
“BITLENGTH” number of shifts are done, and Mode is SHIFT_RIGHT_REPEAT or SHIFT_LEFT_REPEAT, and BITLENGTH >= 32, and Both SIGGENx_DATA0 and SIGGENx_DATA1 have been updated	Copy SIGGENx_DATA1 register contents	Copy SIGGENx_DATA0 register contents
“BITLENGTH” number of shifts are done, and Mode is SHIFT_RIGHT_REPEAT or SHIFT_LEFT_REPEAT, and BITLENGTH <= 32, and Neither SIGGENx_DATA0, nor SIGGENx_DATA1 has been updated	Hold the current value, no shifts	Hold the current value, no shifts
“BITLENGTH” number of shifts are done, and Mode is SHIFT_RIGHT_REPEAT or SHIFT_LEFT_REPEAT, and BITLENGTH >= 32, and Neither SIGGENx_DATA0, nor SIGGENx_DATA1 has not been updated	Hold the current value, no shifts	Hold the current value, no shifts
All other conditions	Updates based on current mode of operation	Updates based on current mode of operation

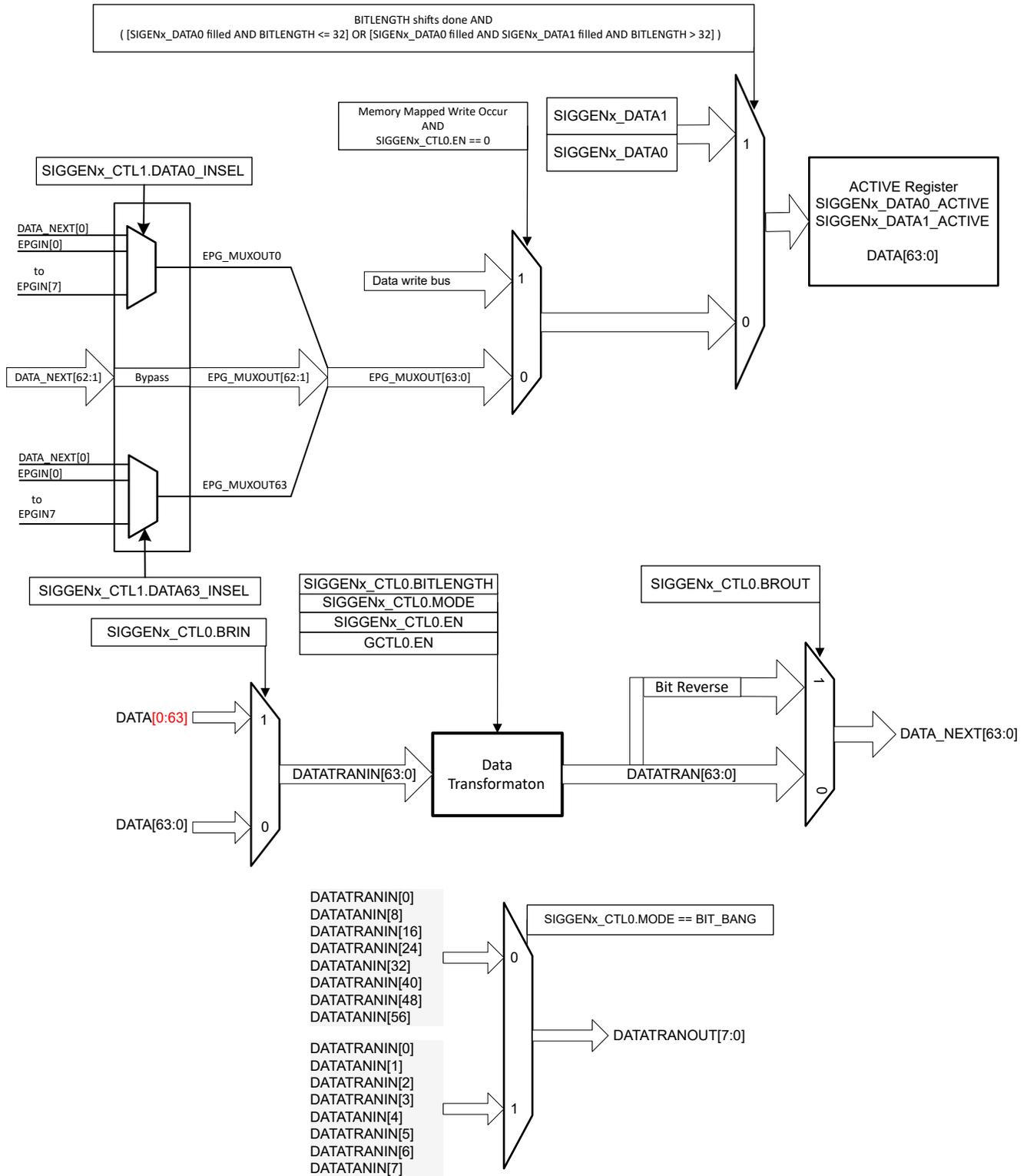


Figure 20-5. EPG Signal Generator Detailed Overview

Following are the possible data transformations:

**Bit-bang mode:** In this mode, DATATRAN[63:0] is the same as DATATRANIN[63:0].

**Shift right once mode:** In this mode, DATATRAN[63:0] = {0,DATATRANIN[63:1]}. After SIGENx\_CTL0.BITLENGTH shifts, SIGENx\_CTL0.EN is cleared.

**Shift right repeat mode:** In this mode, DATATRAN[63:0] = {0,DATATRANIN[63:1]}. After SIGENx\_CTL0.BITLENGTH shifts, load the data as per [Table 20-1](#).

**Rotate right once mode:** In this mode, DATATRAN[63:0] = {DATATRANIN[0],DATATRANIN[63:1]}. After SIGENx\_CTL0.BITLENGTH shifts, active register is loaded from the {DATA1,DATA0} register, and SIGENx\_CTL0.EN is cleared.

**Rotate right repeat:** This mode is same as rotate right once, except that SIGENx\_CTL0.EN is not cleared upon SIGENx\_CTL0.BITLENGTH rotates.

**Shift left once mode:** In this mode, DATATRAN[63:0] = {DATATRANIN[62:1],0}. After SIGENx\_CTL0.BITLENGTH shifts, SIGENx\_CTL0.EN is cleared.

**Shift left repeat mode:** In this mode, DATATRAN[63:0] = {DATATRANIN[62:1],0}. After SIGENx\_CTL0.BITLENGTH shifts, load the data as per [Table 20-1](#).

**Rotate left once mode:** In this mode, DATATRAN[63:0] = {DATATRANIN[62:1],DATATRANIN[63]}. After SIGENx\_CTL0.BITLENGTH shifts, active register is loaded from the {DATA1,DATA0} register, and SIGENx\_CTL0.EN is cleared.

**Rotate left repeat:** This mode is same as rotate left once, except that SIGENx\_CTL0.EN is not cleared upon SIGENx\_CTL0.BITLENGTH rotates.

---

#### Note

SIGGENx\_CTL0.MODE and SIGGENx\_CTL.BITWIDTH must only be updated when SIGGENx\_CTL0.EN is 0.

---

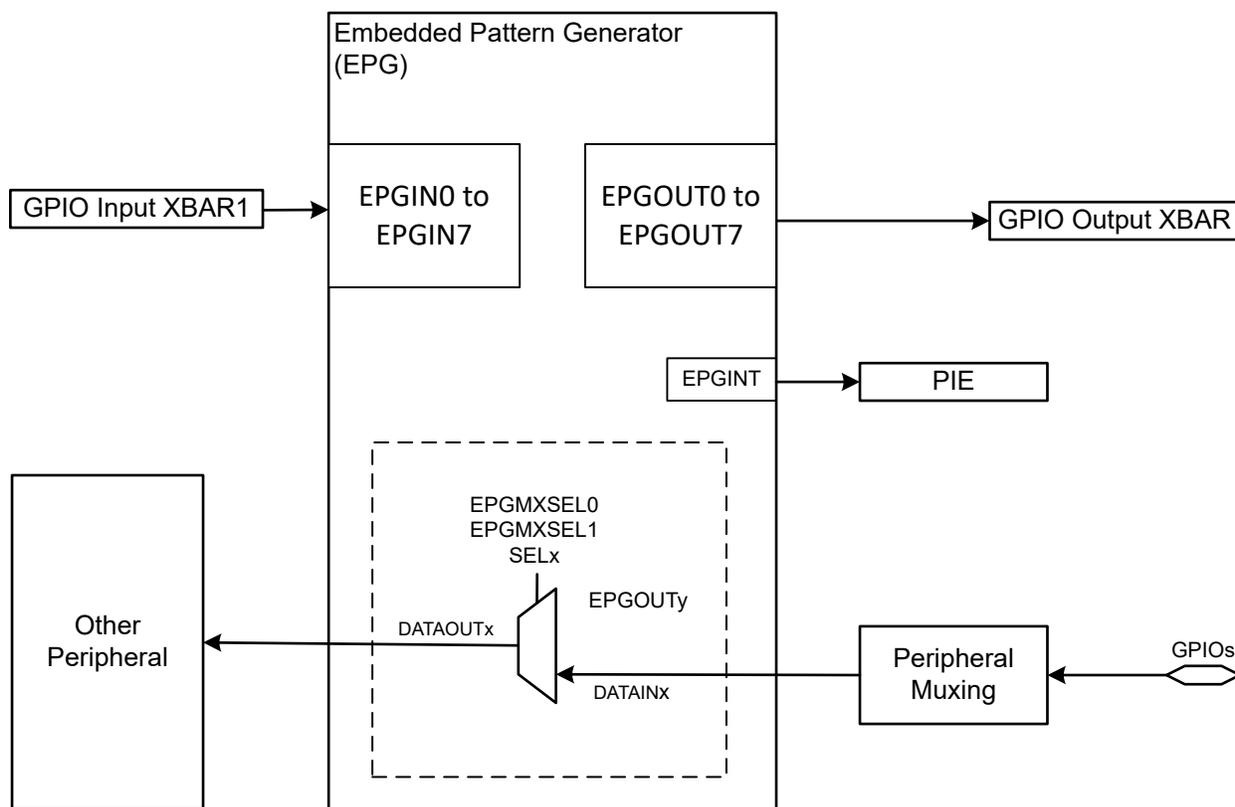
## 20.4 EPG Peripheral Signal Mux Selection

EPG DATAINx signals are connected to input signal sources from other peripherals. The EPG DATAINx signal sources are listed in [Table 20-2](#).

**Table 20-2. EPG Data Input Connections**

EPG1 Data Connection (DATAIN/DATAOUT)	Signal Name
0	CANA_RX
1	MCANA_RX
2	LINA_RX
2-34	Reserved
35	SCIA_RX
36	SCIB_RX
37	SCIC_RX
38	Reserved
39	EQEP1A
40	EQEP1B
41	EQEP1I
42	EQEP1S
43	EQEP2A
44	EQEP2B
45	EQEP2I
46	EQEP2S
47-56	Reserved
57-64	Reserved

EPGOUT0 to EPGOUT7 can override peripheral input signals connected to DATAINx. For example, the peripheral CAN RX input signal can be overwritten by EPGOUTx. DATAINx can be passed to DATAOUTx untouched, if EPG is not required, or the DATAINx can be replaced with EPGOUTy ( $y = x\%8$ ). The EPGMXSEL0 and EPGMXSEL1 registers must be configured to select the source of the DATAOUTx signals. Also, EPGOUTx signals can be connected to GPIOs through the Output XBARs, while EPGINx signals can be connected to the Input XBARs.


**Figure 20-6. EPG Peripheral Signal Muxing**
**Table 20-3. EPG Input Connections**

EPG1 Input Port	Source Signal
0	INPUTXBAR
1	INPUTXBAR
2	INPUTXBAR
3	INPUTXBAR
4	Reserved
5	Reserved
6	Reserved
7	Reserved

**Table 20-4. EPG Output Connections**

EPG1 Output Port	Destination Signal
0	OUTPUTXBAR
1	OUTPUTXBAR
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

## 20.5 EPG Example Use Cases

This section provides example register configurations for different EPG use cases.

### Note

Some examples can require more than one SIGGEN module. Check the device registers or data sheet for the number of available SIGGEN modules.

### 20.5.1 EPG Example: Synchronous Clocks with Offset

[Example 20-1](#) register configuration generates 4 clocks, all synchronous to one another with edges offset by 2 clock cycles. In [Example 20-1](#), a clock divide value of 12 is used.

#### Example 20-1. Synchronous Clocks with Offset Register Configuration

Register	Value	Selected Mode
<b>Epg1MuxRegs</b>		
EPGMXSEL0.SEL0	0x1	Select EPGOUT0 to drive DATAOUT[0]
EPGMXSEL0.SEL1	0x1	Select EPGOUT1 to drive DATAOUT[1]
EPGMXSEL0.SEL2	0x1	Select EPGOUT2 to drive DATAOUT[2]
EPGMXSEL0.SEL3	0x1	Select EPGOUT3 to drive DATAOUT[3]
<b>Epg1Regs</b>		
<b>Global Settings</b>		
GCTL0.EPGOUT0SEL	0x0	Selects signal mux output on EPGOUT0
GCTL0.EPGOUT1SEL	0x0	Selects signal mux output on EPGOUT1
GCTL0.EPGOUT2SEL	0x0	Selects signal mux output on EPGOUT2
GCTL0.EPGOUT3SEL	0x0	Selects signal mux output on EPGOUT3
GCTL3.EPGOUT0_SIGOUTSEL	0x0	Select SIGGEN0.OUT[0] on EPGOUT0
GCTL3.EPGOUT1_SIGOUTSEL	0x1	Select SIGGEN0.OUT[1] on EPGOUT1
GCTL3.EPGOUT2_SIGOUTSEL	0x2	Select SIGGEN0.OUT[2] on EPGOUT2
GCTL3.EPGOUT3_SIGOUTSEL	0x3	Select SIGGEN0.OUT[3] on EPGOUT3
GCTL1.SIGGEN0_CLKSEL	0x0	Select CLKOUT0 of CLKGEN0 as the clock source of SIGGEN0
<b>CLKGEN0 Setting</b>		
CLKDIV0_CTL0.PRD	0x0	Divide by 1
CLKDIV0_CLKOFFSET.CLK0OFFSET	0x0	No offset
<b>SIGGEN0 Mode and Bit Length Configuration</b>		
SIGGEN0_CTL0.BITLENGTH	0xC	Configure bit length to 12 to get a divide-by-12 clock.
SIGGEN0_CTL0.MODE	0x3	Configure the mode to rotate right repeat mode to generate a periodic waveform.
SIGGEN0_DATA0[11:0]	0000 0111 1110	Initialize the 12 bits to 6 ones and 6 zeroes, which makes sure of a 50% duty cycle clock.
SIGGEN0_DATA0[27:16]	0001 1111 1000	Create a 2-cycle offset with respect to first clock.
SIGGEN0_DATA1[11:0]	0111 1110 0000	Create a 4-cycle offset with respect to first clock.
SIGGEN0_DATA1[27:16]	1111 1000 0001	Create a 6-cycle offset with respect to first clock.

## 20.5.2 EPG Example: Serial Data Bit Stream (LSB first)

[Example 20-2](#) register configuration shifts out a data word, the data rate is set to divide by 8 and the LSB is shifted out first. After 32 shifts are completed, an interrupt is generated for further sequencing.

### Example 20-2. Serial Data Bit Stream (LSB first) Register Configuration

Register	Value	Selected Mode
<b>Epg1MuxRegs</b>		
EPGMXSEL0.SEL0	0x1	Select EPGOUT0 to drive DATAOUT[0]
<b>Epg1Regs</b>		
<b>Global Settings</b>		
GCTL1.SIGGEN0_CLKSEL	0x0	Select CLKOUT0 of CLKGEN0 as the clock source of SIGGEN0
<b>CLKGEN0 Setting</b>		
CLKDIV0_CTL0.PRD	0x7	Divide by 8
CLKDIV0_CLKOFFSET.CLK0OFFSET	0x0	No offset
<b>SIGGEN0 Mode and Bit Length Configuration</b>		
SIGGEN0_CTL0.BITLENGTH	0x20	Do 32 shifts.
SIGGEN0_CTL0.MODE	0x1	Configure the mode to shift right once mode. Generates an interrupt after 32 shifts.
SIGGEN0_DATA0[15:0]	0xAA55	Data to be shifted out
SIGGEN0_DATA0[31:16]	0xCCCC	Data to be shifted out
SIGGEN0_DATA1[15:0]	0xAA55	Data to be shifted out
SIGGEN0_DATA1[31:16]	0x55AA	Data to be shifted out

### 20.5.3 EPG Example: Serial Data Bit Stream (MSB first)

[Example 20-3](#) register configuration shifts out a data word, the data rate is set to divide by 8 and MSB is shifted out first. After 32 shifts are complete, an interrupt is generated for further sequencing.

#### Example 20-3. Serial Data Bit Stream (MSB first) Register Configuration

Register	Value	Selected Mode
<b>Epg1MuxRegs</b>		
EPGMXSEL0.SEL0	0x1	Select EPGOUT0 to drive DATAOUT[0]
<b>Epg1Regs</b>		
<b>Global Settings</b>		
GCTL0.EPGOUT0SEL	0x0	Selects signal mux output on EPGOUT0
GCTL3.EPGOUT0_SIGOUTSEL	0x4	Select SIGGEN0.OUT[4] on EPGOUT0, on 64-bit reversal, 31 bit appears on 32 bit (hence, configuring to 4).
GCTL1.SIGGEN0_CLKSEL	0x0	Select CLKOUT0 of CLKGEN0 as the clock source of SIGGEN0
<b>CLKGEN0 Setting</b>		
CLKDIV0_CTL0.PRD	0x7	Divide by 8
CLKDIV0_CLKOFFSET.CLK0OFFSET	0x0	No offset
<b>SIGGEN0 Mode and Bit Length Configuration</b>		
SIGGEN0_CTL0.BITLENGTH	0x20	Do 32 shifts.
SIGGEN0_CTL0.MODE	0x1	Configure the mode to shift right once mode. Generates an interrupt after 32 shifts.
SIGGEN0_CTL0.BRIN	0x1	Reverse the bits to the data transform block to make sure that the MSB is transmitted first.
SIGGEN0_CTL0.BROUT	0x1	Reverse the data back and store in the register
SIGGEN0_DATA0[15:0]	0xAA55	Data to be shifted out
SIGGEN0_DATA0[31:16]	0xCCCC	Data to be shifted out
SIGGEN0_DATA1[15:0]	0xAA55	Data to be shifted out
SIGGEN0_DATA1[31:16]	0x55AA	Data to be shifted out



## 20.7 Software

### 20.7.1 EPG Examples

NOTE: These examples are located in the [C2000Ware](#) installation at the following location:  
C2000Ware\_VERSION#/driverlib/DEVICE\_GPN/examples/CORE\_IF\_MULTICORE/epg

Cloud access to these examples is available at the following link: [dev.ti.com](http://dev.ti.com) [C2000Ware Examples](#).

#### 20.7.1.1 EPG Generating Synchronous Clocks

FILE: epg\_ex1\_generate\_clocks.c

This example shows how to generate 2 synchronous clocks with edges being offset by 2 clock cycles. It configures Signal Generator to shift a periodic data. Generated Clock has period EPG CLOCK/6.

##### *External Connections*

- None. Signal is generated on GPIO 24, 3. Can be visualized through oscilloscope.

##### *Watch Variables*

- sigGenActiveData - Active Data of signal generator transform output

#### 20.7.1.2 EPG Generating Two Offset Clocks

FILE: epg\_ex7\_generate\_two\_offset\_clocks.c

This example generates two offset clocks using the CLKGEN (CLKDIV) modules. For more information on this example, visit: [Designing With the C2000 Embedded Pattern Generator \(EPG\)](#)

##### *External Connections*

- None. Signal is generated on GPIO 24, 3. Can be visualized through oscilloscope.

#### 20.7.1.3 EPG Generating Two Offset Clocks With SIGGEN

FILE: epg\_ex8\_generate\_two\_offset\_clocks\_with\_siggen.c

This example generates two offset clocks using the SIGGEN module. For more information on this example, visit: [Designing With the C2000 Embedded Pattern Generator \(EPG\)](#)

##### *External Connections*

- None. Signal is generated on GPIO 24, 3. Can be visualized through oscilloscope.

#### 20.7.1.4 EPG Generate Serial Data

FILE: epg\_ex9\_generate\_serial\_data.c

This example generates SPICLK and SPI DATA signals using the SIGGEN module. For more information on this example, visit: [Designing With the C2000 Embedded Pattern Generator \(EPG\)](#)

##### *External Connections*

- None. Signal is generated on GPIO 24, 3. Can be visualized through oscilloscope.

#### 20.7.1.5 EPG Generate Serial Data Shift Mode

FILE: epg\_ex10\_generate\_serial\_data\_shift\_mode.c

This example generates SPICLK and SPI DATA signals using the SIGGEN module in SHIFT mode. For more information on this example, visit: [Designing With the C2000 Embedded Pattern Generator \(EPG\)](#)

##### *External Connections*

- None. Signal is generated on GPIO 24, 3. Can be visualized through oscilloscope.

## 20.8 EPG Registers

The following sections are register descriptions for the EPG.

### 20.8.1 EPG Base Address Table

**Table 20-5. EPG Base Address Table**

Bit Field Name		DriverLib Name	Base Address	Pipeline Protected
Instance	Structure			
Epg1Regs	EPG_REGS	EPG1_BASE	0x0005_EC00	YES
Epg1MuxRegs	EPG_MUX_REGS	EPG1MUX_BASE	0x0005_ECD0	YES

## 20.8.2 EPG\_REGS Registers

Table 20-6 lists the memory-mapped registers for the EPG\_REGS registers. All register offset addresses not listed in Table 20-6 should be considered as reserved locations and the register contents should not be modified.

**Table 20-6. EPG\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	GCTL0	EPG Global control register 0		<a href="#">Go</a>
2h	GCTL1	EPG Global control register 1		<a href="#">Go</a>
4h	GCTL2	EPG Global control register 2		<a href="#">Go</a>
6h	GCTL3	EPG Global control register 3		<a href="#">Go</a>
8h	EPGLOCK	EPG LOCK Register		<a href="#">Go</a>
Ah	EPGCOMMIT	EPG COMMIT register		<a href="#">Go</a>
Ch	GINTSTS	EPG Global interrupt status register.		<a href="#">Go</a>
Eh	GINTEN	EPG Global interrupt enable register.		<a href="#">Go</a>
10h	GINTCLR	EPG Global interrupt clear register.		<a href="#">Go</a>
12h	GINTFRC	EPG Global interrupt force register.		<a href="#">Go</a>
18h	CLKDIV0_CTL0	Clock divider 0's control register 0		<a href="#">Go</a>
1Eh	CLKDIV0_CLKOFFSET	Clock divider 0's clock offset value		<a href="#">Go</a>
24h	CLKDIV1_CTL0	Clock divider 1's control register 0		<a href="#">Go</a>
2Ah	CLKDIV1_CLKOFFSET	Clock divider 1's clock offset value		<a href="#">Go</a>
30h	SIGGEN0_CTL0	Signal generator 0's control register 0		<a href="#">Go</a>
32h	SIGGEN0_CTL1	Signal generator 0's control register 1		<a href="#">Go</a>
38h	SIGGEN0_DATA0	Signal generator 0's data register 0		<a href="#">Go</a>
3Ah	SIGGEN0_DATA1	Signal generator 0's data register 1		<a href="#">Go</a>
3Ch	SIGGEN0_DATA0_ACTIVE	Signal generator 0's data active register 0		<a href="#">Go</a>
3Eh	SIGGEN0_DATA1_ACTIVE	Signal generator 0's data active register 1		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 20-7 shows the codes that are used for access types in this section.

**Table 20-7. EPG\_REGS Access Type Codes**

Access Type	Code	Description
Read Type		
R	R	Read
R-0	R-0	Read Returns 0s
Write Type		
W	W	Write
W1C	W1C	Write 1 to clear
W1S	W1S	Write 1 to set
WSONCE	WSONCE	Write Set once
Reset or Default Value		
-n		Value after reset or the default value
Register Array Variables		

**Table 20-7. EPG\_REGS Access Type Codes (continued)**

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

### 20.8.2.1 GCTL0 Register (Offset = 0h) [Reset = 0000000h]

GCTL0 is shown in [Figure 20-8](#) and described in [Table 20-8](#).

Return to the [Summary Table](#).

EPG Global control register 0

**Figure 20-8. GCTL0 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
EPGOUT7SEL	EPGOUT6SEL	EPGOUT5SEL	EPGOUT4SEL	EPGOUT3SEL	EPGOUT2SEL	EPGOUT1SEL	EPGOUT0SEL
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EN
R-0h							R/W-0h

**Table 20-8. GCTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	EPGOUT7SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
14	EPGOUT6SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
13	EPGOUT5SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
12	EPGOUT4SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
11	EPGOUT3SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
10	EPGOUT2SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
9	EPGOUT1SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
8	EPGOUT0SEL	R/W	0h	0 : Selects signal mux output 1 : Selects clock mux output Reset type: SYSRSn
7-1	RESERVED	R	0h	Reserved

**Table 20-8. GCTL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	EN	R/W	0h	0 : EPG module is disabled 1 : EPG module is enabled, clock generators and signal generators are functional. Reset type: SYSRSn

### 20.8.2.2 GCTL1 Register (Offset = 2h) [Reset = 0000000h]

GCTL1 is shown in [Figure 20-9](#) and described in [Table 20-9](#).

Return to the [Summary Table](#).

EPG Global control register 1

**Figure 20-9. GCTL1 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	SIGGEN0_CLKSEL		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		

**Table 20-9. GCTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7	RESERVED	R/W	0h	Reserved
6-4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2-0	SIGGEN0_CLKSEL	R/W	0h	Clock source select of SIGGEN0: 0x0 : CLKGEN0.CLKOUT0_GCLK 0x1 : CLKGEN0.CLKOUT1_GCLK 0x2 : CLKGEN0.CLKOUT2_GCLK 0x3 : CLKGEN0.CLKOUT3_GCLK 0x4 : CLKGEN1.CLKOUT0_GCLK 0x5 : CLKGEN1.CLKOUT1_GCLK 0x6 : CLKGEN1.CLKOUT2_GCLK 0x7 : CLKGEN1.CLKOUT3_GCLK Reset type: SYSRSn

### 20.8.2.3 GCTL2 Register (Offset = 4h) [Reset = 0000000h]

GCTL2 is shown in [Figure 20-10](#) and described in [Table 20-10](#).

Return to the [Summary Table](#).

EPG Global control register 2

**Figure 20-10. GCTL2 Register**

31	30	29	28	27	26	25	24
RESERVED	EPGOUT7_CLKOUTSEL		RESERVED	EPGOUT6_CLKOUTSEL			
R/W-0h	R/W-0h		R/W-0h	R/W-0h			
23	22	21	20	19	18	17	16
RESERVED	EPGOUT5_CLKOUTSEL		RESERVED	EPGOUT4_CLKOUTSEL			
R/W-0h	R/W-0h		R/W-0h	R/W-0h			
15	14	13	12	11	10	9	8
RESERVED	EPGOUT3_CLKOUTSEL		RESERVED	EPGOUT2_CLKOUTSEL			
R/W-0h	R/W-0h		R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	EPGOUT1_CLKOUTSEL		RESERVED	EPGOUT0_CLKOUTSEL			
R/W-0h	R/W-0h		R/W-0h	R/W-0h			

**Table 20-10. GCTL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	Reserved
30-28	EPGOUT7_CLKOUTSEL	R/W	0h	Output 7 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
27	RESERVED	R/W	0h	Reserved
26-24	EPGOUT6_CLKOUTSEL	R/W	0h	Output 6 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
23	RESERVED	R/W	0h	Reserved
22-20	EPGOUT5_CLKOUTSEL	R/W	0h	Output 5 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn

**Table 20-10. GCTL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
19	RESERVED	R/W	0h	Reserved
18-16	EPGOUT4_CLKOUTSEL	R/W	0h	Output 4 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
15	RESERVED	R/W	0h	Reserved
14-12	EPGOUT3_CLKOUTSEL	R/W	0h	Output 3 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
11	RESERVED	R/W	0h	Reserved
10-8	EPGOUT2_CLKOUTSEL	R/W	0h	Output 2 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
7	RESERVED	R/W	0h	Reserved
6-4	EPGOUT1_CLKOUTSEL	R/W	0h	Output 1 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn
3	RESERVED	R/W	0h	Reserved
2-0	EPGOUT0_CLKOUTSEL	R/W	0h	Output 0 signal source select: 0x0 : CLKGEN0.CLKOUT0_DCLK 0x1 : CLKGEN0.CLKOUT1_DCLK 0x2 : CLKGEN0.CLKOUT2_DCLK 0x3 : CLKGEN0.CLKOUT3_DCLK 0x4 : CLKGEN1.CLKOUT0_DCLK 0x5 : CLKGEN1.CLKOUT1_DCLK 0x6 : CLKGEN1.CLKOUT2_DCLK 0x7 : CLKGEN1.CLKOUT3_DCLK Reset type: SYSRSn

### 20.8.2.4 GCTL3 Register (Offset = 6h) [Reset = 0000000h]

GCTL3 is shown in [Figure 20-11](#) and described in [Table 20-11](#).

Return to the [Summary Table](#).

EPG Global control register 3

**Figure 20-11. GCTL3 Register**

31	30	29	28	27	26	25	24
EPGOUT7_SIGOUTSEL				EPGOUT6_SIGOUTSEL			
R/W-0h				R/W-0h			
23	22	21	20	19	18	17	16
EPGOUT5_SIGOUTSEL				EPGOUT4_SIGOUTSEL			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
EPGOUT3_SIGOUTSEL				EPGOUT2_SIGOUTSEL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
EPGOUT1_SIGOUTSEL				EPGOUT0_SIGOUTSEL			
R/W-0h				R/W-0h			

**Table 20-11. GCTL3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	EPGOUT7_SIGOUTSEL	R/W	0h	Output 7 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn

**Table 20-11. GCTL3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27-24	EPGOUT6_SIGOUTSEL	R/W	0h	Output 6 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn
23-20	EPGOUT5_SIGOUTSEL	R/W	0h	Output 5 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn
19-16	EPGOUT4_SIGOUTSEL	R/W	0h	Output 4 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn

**Table 20-11. GCTL3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-12	EPGOUT3_SIGOUTSEL	R/W	0h	Output 3 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn
11-8	EPGOUT2_SIGOUTSEL	R/W	0h	Output 2 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn
7-4	EPGOUT1_SIGOUTSEL	R/W	0h	Output 1 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn

**Table 20-11. GCTL3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	EPGOUT0_SIGOUTSEL	R/W	0h	Output 0 source select: 0x0 : SIGGEN0.DATATRANOUT0 0x1 : SIGGEN0.DATATRANOUT1 0x2 : SIGGEN0.DATATRANOUT2 0x3 : SIGGEN0.DATATRANOUT3 0x4 : SIGGEN0.DATATRANOUT4 0x5 : SIGGEN0.DATATRANOUT5 0x6 : SIGGEN0.DATATRANOUT6 0x7 : SIGGEN0.DATATRANOUT7 0x8 : SIGGEN1.DATATRANOUT0 0x9 : SIGGEN1.DATATRANOUT1 0xA : SIGGEN1.DATATRANOUT2 0xB : SIGGEN1.DATATRANOUT3 0xC : SIGGEN1.DATATRANOUT4 0xD : SIGGEN1.DATATRANOUT5 0xE : SIGGEN1.DATATRANOUT6 0xF : SIGGEN1.DATATRANOUT7 Reset type: SYSRSn

### 20.8.2.5 EPGLOCK Register (Offset = 8h) [Reset = 0000000h]

EPGLOCK is shown in [Figure 20-12](#) and described in [Table 20-12](#).

Return to the [Summary Table](#).

EPG LOCK Register

**Figure 20-12. EPGLOCK 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/W-0h	R/W-0h
7	6	5	4	3	2	1	0
SIGGEN0_CTL1	SIGGEN0_CTL0	CLKDIV1_CTL0	CLKDIV0_CTL0	GCTL3	GCTL2	GCTL1	GCTL0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 20-12. EPGLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	RESERVED	R/W	0h	Reserved
8	RESERVED	R/W	0h	Reserved
7	SIGGEN0_CTL1	R/W	0h	0: Writes to SIGGEN0_CTL1 register is allowed. 1: Writes to SIGGEN0_CTL1 register is not allowed. Reset type: SYSRSn
6	SIGGEN0_CTL0	R/W	0h	0: Writes to SIGGEN0_CTL0 register is allowed. 1: Writes to SIGGEN0_CTL0 register is not allowed. Reset type: SYSRSn
5	CLKDIV1_CTL0	R/W	0h	0: Writes to CLKDIV1_CTL0 register is allowed. 1: Writes to CLKDIV1_CTL0 register is not allowed. Reset type: SYSRSn
4	CLKDIV0_CTL0	R/W	0h	0: Writes to CLKDIV0_CTL0 register is allowed. 1: Writes to CLKDIV0_CTL0 register is not allowed. Reset type: SYSRSn
3	GCTL3	R/W	0h	0: Writes to GCTL3 register is allowed. 1: Writes to GCTL3 register is not allowed. Reset type: SYSRSn
2	GCTL2	R/W	0h	0: Writes to GCTL2 register is allowed. 1: Writes to GCTL2 register is not allowed. Reset type: SYSRSn
1	GCTL1	R/W	0h	0: Writes to GCTL1 register is allowed. 1: Writes to GCTL1 register is not allowed. Reset type: SYSRSn
0	GCTL0	R/W	0h	0: Writes to GCTL0 register is allowed. 1: Writes to GCTL0 register is not allowed. Reset type: SYSRSn

### 20.8.2.6 EPGCOMMIT Register (Offset = Ah) [Reset = 0000000h]

EPGCOMMIT is shown in [Figure 20-13](#) and described in [Table 20-13](#).

Return to the [Summary Table](#).

EPG COMMIT register

**Figure 20-13. EPGCOMMIT 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/WOnce-0h	R/WOnce-0h
7	6	5	4	3	2	1	0
SIGGEN0_CTL 1	SIGGEN0_CTL 0	CLKDIV1_CTL0	CLKDIV0_CTL0	GCTL3	GCTL2	GCTL1	GCTL0
R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h	R/WOnce-0h

**Table 20-13. EPGCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	RESERVED	R/WOnce	0h	Reserved
8	RESERVED	R/WOnce	0h	Reserved
7	SIGGEN0_CTL1	R/WOnce	0h	0: Writes to EPGLOCK.SIGGEN0_CTL1 field is allowed. 1: Writes to EPGLOCK.SIGGEN0_CTL1 field is not allowed. Reset type: SYSRSn
6	SIGGEN0_CTL0	R/WOnce	0h	0: Writes to EPGLOCK.SIGGEN0_CTL0 field is allowed. 1: Writes to EPGLOCK.SIGGEN0_CTL0 field is not allowed. Reset type: SYSRSn
5	CLKDIV1_CTL0	R/WOnce	0h	0: Writes to EPGLOCK.CLKDIV1_CTL0 field is allowed. 1: Writes to EPGLOCK.CLKDIV1_CTL0 field is not allowed. Reset type: SYSRSn
4	CLKDIV0_CTL0	R/WOnce	0h	0: Writes to EPGLOCK.CLKDIV0_CTL0 field is allowed. 1: Writes to EPGLOCK.CLKDIV0_CTL0 field is not allowed. Reset type: SYSRSn
3	GCTL3	R/WOnce	0h	0: Writes to EPGLOCK.GCTL3 field is allowed. 1: Writes to EPGLOCK.GCTL3 field is not allowed. Reset type: SYSRSn
2	GCTL2	R/WOnce	0h	0: Writes to EPGLOCK.GCTL2 field is allowed. 1: Writes to EPGLOCK.GCTL2 field is not allowed. Reset type: SYSRSn
1	GCTL1	R/WOnce	0h	0: Writes to EPGLOCK.GCTL1 field is allowed. 1: Writes to EPGLOCK.GCTL1 field is not allowed. Reset type: SYSRSn
0	GCTL0	R/WOnce	0h	0: Writes to EPGLOCK.GCTL0 field is allowed. 1: Writes to EPGLOCK.GCTL0 field is not allowed. Reset type: SYSRSn

### 20.8.2.7 GINTSTS Register (Offset = Ch) [Reset = 0000000h]

GINTSTS is shown in [Figure 20-14](#) and described in [Table 20-14](#).

Return to the [Summary Table](#).

EPG Global interrupt status register.

**Figure 20-14. GINTSTS 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	SIGGEN0_FILL	SIGGEN0_DON E	INT
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h

**Table 20-14. GINTSTS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R	0h	Reserved
3	RESERVED	R	0h	Reserved
2	SIGGEN0_FILL	R	0h	0: Do not fill data in SIGGEN0 1: Fill data in SIGGEN0 This status bit does not get set in BIT_BANG mode. In all other modes, the SIGGEN0_FILL bit is set high after the signal generator has completed BITLENGTH/2 shifts. Note: For odd values of BITLENGTH, BITLENGTH/2 is rounded down to the nearest integer. Reset type: SYSRSn
1	SIGGEN0_DONE	R	0h	0: Operation of SIGGEN0 is in progress 1: Operation of SIGGEN0 has completed This status bit does not get set in BIT_BANG mode. In all other modes, the SIGGEN0_DONE bit is set high after the signal generator has completed BITLENGTH shifts. Reset type: SYSRSn
0	INT	R	0h	Global interrupt flag. This bit is set when an interrupt is fired, and cleared by writing 1 to GINTCLR.INT. While the INT status bit is set, new EPG interrupts cannot be generated until the bit has been cleared. Reset type: SYSRSn

### 20.8.2.8 GINTEN Register (Offset = Eh) [Reset = 0000000h]

GINTEN is shown in [Figure 20-15](#) and described in [Table 20-15](#).

Return to the [Summary Table](#).

EPG Global interrupt enable register.

**Figure 20-15. GINTEN 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	SIGGEN0_FILL	SIGGEN0_DON E	RESERVED
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

**Table 20-15. GINTEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R/W	0h	Reserved
3	RESERVED	R/W	0h	Reserved
2	SIGGEN0_FILL	R/W	0h	0: Disable interrupt generation when SIGGEN0_FILL bits is set. 1: Enable interrupt generation when SIGGEN0_FILL bits is set. Reset type: SYSRSn
1	SIGGEN0_DONE	R/W	0h	0: Disable interrupt generation when SIGGEN0_DONE bits is set. 1: Enable interrupt generation when SIGGEN0_DONE bits is set. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

### 20.8.2.9 GINTCLR Register (Offset = 10h) [Reset = 0000000h]

GINTCLR is shown in [Figure 20-16](#) and described in [Table 20-16](#).

Return to the [Summary Table](#).

EPG Global interrupt clear register.

**Figure 20-16. GINTCLR 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	SIGGEN0_FILL	SIGGEN0_DON E	INT
R-0h			R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h	R-0/W1C-0h

**Table 20-16. GINTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R-0/W1C	0h	Reserved
3	RESERVED	R-0/W1C	0h	Reserved
2	SIGGEN0_FILL	R-0/W1C	0h	0: No effect 1: Clear SIGGEN0_FILL flag bit. Reset type: SYSRSn
1	SIGGEN0_DONE	R-0/W1C	0h	0: No effect 1: Clear SIGGEN0_DONE flag bit. Reset type: SYSRSn
0	INT	R-0/W1C	0h	0: No effect 1: Clear INT flag bit. Reset type: SYSRSn

### 20.8.2.10 GINTFRC Register (Offset = 12h) [Reset = 0000000h]

GINTFRC is shown in [Figure 20-17](#) and described in [Table 20-17](#).

Return to the [Summary Table](#).

EPG Global interrupt force register.

**Figure 20-17. GINTFRC 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	SIGGEN0_FILL	SIGGEN0_DON E	RESERVED
R-0h			R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0/W1S-0h	R-0h

**Table 20-17. GINTFRC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RESERVED	R-0/W1S	0h	Reserved
3	RESERVED	R-0/W1S	0h	Reserved
2	SIGGEN0_FILL	R-0/W1S	0h	0: No effect 1: set SIGGEN0_FILL flag bit. Reset type: SYSRSn
1	SIGGEN0_DONE	R-0/W1S	0h	0: No effect 1: set SIGGEN0_DONE flag bit. Reset type: SYSRSn
0	RESERVED	R	0h	Reserved

### 20.8.2.11 CLKDIV0\_CTL0 Register (Offset = 18h) [Reset = 0000000h]

CLKDIV0\_CTL0 is shown in [Figure 20-18](#) and described in [Table 20-18](#).

Return to the [Summary Table](#).

Clock divider 0's control register 0

**Figure 20-18. CLKDIV0\_CTL0 Register**

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

**Table 20-18. CLKDIV0\_CTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	CLKSTOP	R/W	0h	Determines on which of the CLKOUTs edge clock generation is stopped following a clear of SIGGEN0_CTL0.EN. 000 : Stop on CLKOUT0 010 : Stop on CLKOUT1 100 : Stop on CLKOUT2 110 : Stop on CLKOUT3 Reset type: SYSRSn
15-8	RESERVED	R	0h	Reserved
7-0	PRD	R/W	0h	Clock divider period: Clock divider counter counts up to period (PRD) and snaps back to 0. Reset type: SYSRSn

### 20.8.2.12 CLKDIV0\_CLKOFFSET Register (Offset = 1Eh) [Reset = 0000000h]

CLKDIV0\_CLKOFFSET is shown in [Figure 20-19](#) and described in [Table 20-19](#).

Return to the [Summary Table](#).

Clock divider 0's clock offset value

**Figure 20-19. CLKDIV0\_CLKOFFSET Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLK3OFFSET								CLK2OFFSET							
R/W-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK1OFFSET								CLK0OFFSET							
R/W-0h								R/W-0h							

**Table 20-19. CLKDIV0\_CLKOFFSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	CLK3OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 3 (CLKOUT3) is delayed. Reset type: SYSRSn
23-16	CLK2OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 2 (CLKOUT2) is delayed. Reset type: SYSRSn
15-8	CLK1OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 1 (CLKOUT1) is delayed. Reset type: SYSRSn
7-0	CLK0OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 0 (CLKOUT0) is delayed. Reset type: SYSRSn

### 20.8.2.13 CLKDIV1\_CTL0 Register (Offset = 24h) [Reset = 0000000h]

CLKDIV1\_CTL0 is shown in [Figure 20-20](#) and described in [Table 20-20](#).

Return to the [Summary Table](#).

Clock divider 1's control register 0

**Figure 20-20. CLKDIV1\_CTL0 Register**

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

**Table 20-20. CLKDIV1\_CTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Reserved
18-16	CLKSTOP	R/W	0h	Determines on which of the CLKOUTs edge clock generation is stopped following a clear of SIGGEN1_CTL0.EN. 000 : Stop on CLKOUT0 010 : Stop on CLKOUT1 100 : Stop on CLKOUT2 110 : Stop on CLKOUT3 Reset type: SYSRSn
15-8	RESERVED	R	0h	Reserved
7-0	PRD	R/W	0h	Clock divider period: Clock divider counter counts up to period (PRD) and snaps back to 0. Reset type: SYSRSn

### 20.8.2.14 CLKDIV1\_CLKOFFSET Register (Offset = 2Ah) [Reset = 0000000h]

CLKDIV1\_CLKOFFSET is shown in [Figure 20-21](#) and described in [Table 20-21](#).

Return to the [Summary Table](#).

Clock divider 1's clock offset value

**Figure 20-21. CLKDIV1\_CLKOFFSET Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLK3OFFSET								CLK2OFFSET							
R/W-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK1OFFSET								CLK0OFFSET							
R/W-0h								R/W-0h							

**Table 20-21. CLKDIV1\_CLKOFFSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	CLK3OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 3 (CLKOUT3) is delayed. Reset type: SYSRSn
23-16	CLK2OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 2 (CLKOUT2) is delayed. Reset type: SYSRSn
15-8	CLK1OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 1 (CLKOUT1) is delayed. Reset type: SYSRSn
7-0	CLK0OFFSET	R/W	0h	Number of source clock cycles by which the divided clock output 0 (CLKOUT0) is delayed. Reset type: SYSRSn

### 20.8.2.15 SIGGEN0\_CTL0 Register (Offset = 30h) [Reset = 0000000h]

SIGGEN0\_CTL0 is shown in [Figure 20-22](#) and described in [Table 20-22](#).

Return to the [Summary Table](#).

Signal generator 0's control register 0

**Figure 20-22. SIGGEN0\_CTL0 Register**

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

**Table 20-22. SIGGEN0\_CTL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-16	BITLENGTH	R/W	0h	Defines the number bits which participates in the shift rotate operations. Reset type: SYSRSn
15-7	RESERVED	R	0h	Reserved
6	BROUT	R/W	0h	0 : No bit reversal on data output from data transform block 1 : Perform bit reversal on data output from data transform block Reset type: SYSRSn
5	BRIN	R/W	0h	0 : No bit reversal on data input of data transform block 1 : Perform bit reversal on data input of data transform block Reset type: SYSRSn
4	EN	R/W	0h	0 : Signal generator is disabled. 1 : Signal generator is enabled, signal generator functions as per the mode definition. Reset type: SYSRSn

**Table 20-22. SIGGEN0\_CTL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	MODE	R/W	0h	<p>0 : BIT_BANG mode, The value written into DATA0 and DATA1 registers appear on the signal generator outputs as is.</p> <p>1 : SHIFT_RIGHT_ONCE mode, The data value written into (DATA1,DATA0) registers are shifted right by 1 on every clock. Shifting operations stops when BITLENGTH shifts are done and SIGGEN0_CTL0.EN bit is cleared.</p> <p>2 : ROTATE_RIGHT_ONCE, The data value written into (DATA1,DATA0) registers are rotated right by 1 on every clock. Rotation happens within (DATA1,DATA0)[BITLENGTH-1:0] Rotate operations stops when BITLENGTH shifts are done and SIGGEN0_CTL0.EN bit is cleared.</p> <p>3 : ROTATE_RIGHT_REPEAT, The data value written into (DATA1,DATA0) registers are rotated right by 1 on every clock. Rotation happens within (DATA1,DATA0)[BITLENGTH-1:0] Rotate operations continue until SIGGEN0_CTL0.EN bit is cleared.</p> <p>4 : SHIFT_LEFT_ONCE mode, The data value written into (DATA1,DATA0) registers are shifted left by 1 on every clock. Shifting operations stops when BITLENGTH shifts are done and SIGGEN0_CTL0.EN bit is cleared.</p> <p>5 : ROTATE_LEFT_ONCE, The data value written into (DATA1,DATA0) registers are rotated left by 1 on every clock. Rotation happens within (DATA1,DATA0)[BITLENGTH-1:0] Rotate operations stops when BITLENGTH shifts are done and SIGGEN0_CTL0.EN bit is cleared.</p> <p>6 : ROTATE_LEFT_REPEAT, The data value written into (DATA1,DATA0) registers are rotated left by 1 on every clock. Rotation happens within (DATA1,DATA0)[BITLENGTH-1:0] Rotate operations continue until SIGGEN0_CTL0.EN bit is cleared.</p> <p>7 : SHIFT_RIGHT_REPEAT mode, The data value written into (DATA1,DATA0) registers are shifted right by 1 on every clock. Shifting operations stops when SIGGEN0_CTL0.EN bit is cleared.</p> <p>8 : SHIFT_LEFT_REPEAT mode, The data value written into (DATA1,DATA0) registers are shifted left by 1 on every clock. Shifting operations stops when SIGGEN0_CTL0.EN bit is cleared.</p> <p>Reset type: SYSRSn</p>

### 20.8.2.16 SIGGEN0\_CTL1 Register (Offset = 32h) [Reset = 0000000h]

SIGGEN0\_CTL1 is shown in [Figure 20-23](#) and described in [Table 20-23](#).

Return to the [Summary Table](#).

Signal generator 0's control register 1

**Figure 20-23. SIGGEN0\_CTL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA63_INSEL				RESERVED											
R/W-0h				R-0h											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												DATA0_INSEL			
R-0h												R/W-0h			

**Table 20-23. SIGGEN0\_CTL1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	DATA63_INSEL	R/W	0h	Source input of bit 63 of Data register. If 0 selects DATA_NEXT[63] else, selects one of the EPGIN inputs. This provides the ability to capture the data. 0x0 : DATA_NEXT[63] 0x1 : EPGIN0 0x2 : EPGIN1 0x3 : EPGIN2 0x4 : EPGIN3 0x5 : EPGIN4 0x6 : EPGIN5 0x7 : EPGIN6 0x8 : EPGIN7 0x9-0xF : 0 Reset type: SYSRSn
27-4	RESERVED	R	0h	Reserved
3-0	DATA0_INSEL	R/W	0h	Source input of bit 0 of Data register. If 0 selects DATA_NEXT[0] else, selects one of the EPGIN inputs. This provides the ability to capture the data. 0x0 : DATA_NEXT[0] 0x1 : EPGIN0 0x2 : EPGIN1 0x3 : EPGIN2 0x4 : EPGIN3 0x5 : EPGIN4 0x6 : EPGIN5 0x7 : EPGIN6 0x8 : EPGIN7 0x9-0xF : 0 Reset type: SYSRSn

### 20.8.2.17 SIGGEN0\_DATA0 Register (Offset = 38h) [Reset = 00000000h]

SIGGEN0\_DATA0 is shown in [Figure 20-24](#) and described in [Table 20-24](#).

Return to the [Summary Table](#).

Signal generator 0's data register 0

**Figure 20-24. SIGGEN0\_DATA0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGGEN_DATA0																															
R/W-0h																															

**Table 20-24. SIGGEN0\_DATA0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGGEN_DATA0	R/W	0h	Data used in signal bit stream. {SIGGEN_DATA1,SIGGEN_DATA0} together constitutes a 64 bit data stream, which are modified as per the SIGGENx_CTL0.MODE configuration. Reset type: SYSRSn

### 20.8.2.18 SIGGEN0\_DATA1 Register (Offset = 3Ah) [Reset = 0000000h]

SIGGEN0\_DATA1 is shown in [Figure 20-25](#) and described in [Table 20-25](#).

Return to the [Summary Table](#).

Signal generator 0's data register 1

**Figure 20-25. SIGGEN0\_DATA1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGGEN_DATA1																															
R/W-0h																															

**Table 20-25. SIGGEN0\_DATA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGGEN_DATA1	R/W	0h	Data used in signal bit stream. {SIGGEN_DATA1,SIGGEN_DATA0} together constitutes a 64 bit data stream, which are modified as per the SIGGENx_CTL0.MODE configuration. Reset type: SYSRSn

**20.8.2.19 SIGGEN0\_DATA0\_ACTIVE Register (Offset = 3Ch) [Reset = 0000000h]**

SIGGEN0\_DATA0\_ACTIVE is shown in [Figure 20-26](#) and described in [Table 20-26](#).

Return to the [Summary Table](#).

Signal generator 0's data active register 0

**Figure 20-26. SIGGEN0\_DATA0\_ACTIVE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGEN_DATA0																															
R-0h																															

**Table 20-26. SIGGEN0\_DATA0\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGEN_DATA0	R	0h	This is the lower 32 bits of the 64 bit active register (used in data transformation) Reset type: SYSRSn

### 20.8.2.20 SIGGEN0\_DATA1\_ACTIVE Register (Offset = 3Eh) [Reset = 0000000h]

SIGGEN0\_DATA1\_ACTIVE is shown in [Figure 20-27](#) and described in [Table 20-27](#).

Return to the [Summary Table](#).

Signal generator 0's data active register 1

**Figure 20-27. SIGGEN0\_DATA1\_ACTIVE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGGEN_DATA1																															
R-0h																															

**Table 20-27. SIGGEN0\_DATA1\_ACTIVE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGGEN_DATA1	R	0h	This is the upper 32 bits of the 64 bit active register (used in data transformation) Reset type: SYSRSn

### 20.8.3 EPG\_MUX\_REGS Registers

Table 20-28 lists the memory-mapped registers for the EPG\_MUX\_REGS registers. All register offset addresses not listed in Table 20-28 should be considered as reserved locations and the register contents should not be modified.

**Table 20-28. EPG\_MUX\_REGS Registers**

Offset	Acronym	Register Name	Write Protection	Section
0h	EPGMXSEL0	EPG Mux select register 0		<a href="#">Go</a>
Ch	EPGMXSELLOCK	EPG Mux select register lock		<a href="#">Go</a>
Eh	EPGMXSELCOMMIT	EPG Mux select register commit		<a href="#">Go</a>

Complex bit access types are encoded to fit into small table cells. Table 20-29 shows the codes that are used for access types in this section.

**Table 20-29. EPG\_MUX\_REGS Access Type Codes**

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

### 20.8.3.1 EPGMXSEL0 Register (Offset = 0h) [Reset = 0000000h]

EPGMXSEL0 is shown in [Figure 20-28](#) and described in [Table 20-30](#).

Return to the [Summary Table](#).

EPG Mux select register 0

**Figure 20-28. EPGMXSEL0 Register**

31	30	29	28	27	26	25	24
SEL31	SEL30	SEL29	SEL28	SEL27	SEL26	SEL25	SEL24
R/W-0h							
23	22	21	20	19	18	17	16
SEL23	SEL22	SEL21	SEL20	SEL19	SEL18	SEL17	SEL16
R/W-0h							
15	14	13	12	11	10	9	8
SEL15	SEL14	SEL13	SEL12	SEL11	SEL10	SEL9	SEL8
R/W-0h							
7	6	5	4	3	2	1	0
SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0
R/W-0h							

**Table 20-30. EPGMXSEL0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	SEL31	R/W	0h	0: DATAIN[31] is selected 1: EPGOUT[7] is selected Reset type: SYSRSn
30	SEL30	R/W	0h	0: DATAIN[30] is selected 1: EPGOUT[6] is selected Reset type: SYSRSn
29	SEL29	R/W	0h	0: DATAIN[29] is selected 1: EPGOUT[5] is selected Reset type: SYSRSn
28	SEL28	R/W	0h	0: DATAIN[28] is selected 1: EPGOUT[4] is selected Reset type: SYSRSn
27	SEL27	R/W	0h	0: DATAIN[27] is selected 1: EPGOUT[3] is selected Reset type: SYSRSn
26	SEL26	R/W	0h	0: DATAIN[26] is selected 1: EPGOUT[2] is selected Reset type: SYSRSn
25	SEL25	R/W	0h	0: DATAIN[25] is selected 1: EPGOUT[1] is selected Reset type: SYSRSn
24	SEL24	R/W	0h	0: DATAIN[24] is selected 1: EPGOUT[0] is selected Reset type: SYSRSn
23	SEL23	R/W	0h	0: DATAIN[23] is selected 1: EPGOUT[7] is selected Reset type: SYSRSn
22	SEL22	R/W	0h	0: DATAIN[22] is selected 1: EPGOUT[6] is selected Reset type: SYSRSn

**Table 20-30. EPGMXSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
21	SEL21	R/W	0h	0: DATAIN[21] is selected 1: EPGOUT[5] is selected Reset type: SYSRSn
20	SEL20	R/W	0h	0: DATAIN[20] is selected 1: EPGOUT[4] is selected Reset type: SYSRSn
19	SEL19	R/W	0h	0: DATAIN[19] is selected 1: EPGOUT[3] is selected Reset type: SYSRSn
18	SEL18	R/W	0h	0: DATAIN[18] is selected 1: EPGOUT[2] is selected Reset type: SYSRSn
17	SEL17	R/W	0h	0: DATAIN[17] is selected 1: EPGOUT[1] is selected Reset type: SYSRSn
16	SEL16	R/W	0h	0: DATAIN[16] is selected 1: EPGOUT[0] is selected Reset type: SYSRSn
15	SEL15	R/W	0h	0: DATAIN[15] is selected 1: EPGOUT[7] is selected Reset type: SYSRSn
14	SEL14	R/W	0h	0: DATAIN[14] is selected 1: EPGOUT[6] is selected Reset type: SYSRSn
13	SEL13	R/W	0h	0: DATAIN[13] is selected 1: EPGOUT[5] is selected Reset type: SYSRSn
12	SEL12	R/W	0h	0: DATAIN[12] is selected 1: EPGOUT[4] is selected Reset type: SYSRSn
11	SEL11	R/W	0h	0: DATAIN[11] is selected 1: EPGOUT[3] is selected Reset type: SYSRSn
10	SEL10	R/W	0h	0: DATAIN[10] is selected 1: EPGOUT[2] is selected Reset type: SYSRSn
9	SEL9	R/W	0h	0: DATAIN[9] is selected 1: EPGOUT[1] is selected Reset type: SYSRSn
8	SEL8	R/W	0h	0: DATAIN[8] is selected 1: EPGOUT[0] is selected Reset type: SYSRSn
7	SEL7	R/W	0h	0: DATAIN[7] is selected 1: EPGOUT[7] is selected Reset type: SYSRSn
6	SEL6	R/W	0h	0: DATAIN[6] is selected 1: EPGOUT[6] is selected Reset type: SYSRSn
5	SEL5	R/W	0h	0: DATAIN[5] is selected 1: EPGOUT[5] is selected Reset type: SYSRSn

**Table 20-30. EPGMXSEL0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	SEL4	R/W	0h	0: DATAIN[4] is selected 1: EPGOUT[4] is selected Reset type: SYSRSn
3	SEL3	R/W	0h	0: DATAIN[3] is selected 1: EPGOUT[3] is selected Reset type: SYSRSn
2	SEL2	R/W	0h	0: DATAIN[2] is selected 1: EPGOUT[2] is selected Reset type: SYSRSn
1	SEL1	R/W	0h	0: DATAIN[1] is selected 1: EPGOUT[1] is selected Reset type: SYSRSn
0	SEL0	R/W	0h	0: DATAIN[0] is selected 1: EPGOUT[0] is selected Reset type: SYSRSn

### 20.8.3.2 EPGMXSELLOCK Register (Offset = Ch) [Reset = 0000000h]

EPGMXSELLOCK is shown in [Figure 20-29](#) and described in [Table 20-31](#).

Return to the [Summary Table](#).

EPG Mux select register lock

**Figure 20-29. EPGMXSELLOCK 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	EPGMXSEL0
R-0h						R/W-0h	R/W-0h

**Table 20-31. EPGMXSELLOCK Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	RESERVED	R/W	0h	Reserved
0	EPGMXSEL0	R/W	0h	0: Writes to EPGMXSEL0 registers are allowed. 1: Writes to EPGMXSEL0 registers are not allowed. Reset type: SYSRSn

### 20.8.3.3 EPGMXSELCOMMIT Register (Offset = Eh) [Reset = 0000000h]

EPGMXSELCOMMIT is shown in [Figure 20-30](#) and described in [Table 20-32](#).

Return to the [Summary Table](#).

EPG Mux select register commit

**Figure 20-30. EPGMXSELCOMMIT 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	EPGMXSEL0
R-0h						R/WOnce-0h	R/WOnce-0h

**Table 20-32. EPGMXSELCOMMIT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved
1	RESERVED	R/WOnce	0h	Reserved
0	EPGMXSEL0	R/WOnce	0h	0: Writes to EPGMXSELLOCK.EPGMXSEL12 field is allowed. 1: Writes to EPGMXSELLOCK.EPGMXSEL12 field is not allowed. Reset type: SYSRSn

### 20.8.4 EPG Registers to Driverlib Functions

**Table 20-33. EPG Registers to Driverlib Functions**

File	Driverlib Function
<b>GCTL0</b>	
epg.h	EPG_enableGlobal
epg.h	EPG_disableGlobal
epg.h	EPG_selectEPGOutput
<b>GCTL1</b>	
epg.h	EPG_selectSigGenClkSource
<b>GCTL2</b>	
epg.h	EPG_selectClkOutput
<b>GCTL3</b>	
epg.h	EPG_selectSignalOutput
<b>LOCK</b>	
epg.h	EPG_lockReg
epg.h	EPG_releaseLockReg
<b>COMMIT</b>	
epg.h	EPG_commitRegLock
<b>GINTSTS</b>	
epg.h	EPG_getInterruptStatus

**Table 20-33. EPG Registers to Driverlib Functions (continued)**

File	Driverlib Function
<b>GINTEN</b>	
epg.h	EPG_enableInterruptFlag
epg.h	EPG_disableInterruptFlag
<b>GINTCLR</b>	
epg.h	EPG_clearInterruptFlag
<b>GINTFRC</b>	
epg.h	EPG_forceInterruptFlag
<b>CLKDIV0_CTL0</b>	
epg.h	EPG_setClkGenPeriod
epg.h	EPG_setClkGenStopEdge
<b>CLKDIV0_CLKOFFSET</b>	
epg.h	EPG_setClkGenOffset
<b>CLKDIV1_CTL0</b>	
-	See CLKDIV0_CTL0
<b>CLKDIV1_CLKOFFSET</b>	
-	See CLKDIV0_CLKOFFSET
<b>SIGGEN0_CTL0</b>	
epg.h	EPG_enableSignalGen
epg.h	EPG_disableSignalGen
epg.h	EPG_setSignalGenMode
epg.h	EPG_enableBitRevOnDataIn
epg.h	EPG_disableBitRevOnDataIn
epg.h	EPG_enableBitRevOnDataOut
epg.h	EPG_disableBitRevOnDataOut
epg.h	EPG_setDataBitLen
<b>SIGGEN0_CTL1</b>	
epg.h	EPG_setData0In
epg.h	EPG_setData63In
<b>SIGGEN0_DATA0</b>	
epg.h	EPG_setData0Word
epg.h	EPG_getData0ActiveReg
<b>SIGGEN0_DATA1</b>	
epg.h	EPG_setData1Word
epg.h	EPG_getData1ActiveReg
<b>SIGGEN0_DATA0_ACTIVE</b>	
epg.h	EPG_getData0ActiveReg
<b>SIGGEN0_DATA1_ACTIVE</b>	
epg.h	EPG_getData1ActiveReg
<b>MXSEL0</b>	
epg.c	EPG_selectEPGDataOut
<b>MXSELLOCK</b>	
epg.h	EPG_lockMXSelReg
epg.h	EPG_releaseLockMXSelReg
<b>MXSELCOMMIT</b>	

**Table 20-33. EPG Registers to Driverlib Functions (continued)**

File	Driverlib Function
epg.h	EPG_commitMXSelRegLock

# Revision History



NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

## Changes from March 16, 2023 to April 2, 2024 (from Revision A (March 2023) to Revision B (April 2024))

	Page
• Deleted <i>VCRC Unit</i> section in <a href="#">Chapter 2</a> .....	57
• Changed Name of Vector ID 17 in <a href="#">Table 3-4</a> .....	73
• Changed <a href="#">Section 3.7.13</a> .....	92
• Deleted <i>PLL Reference Clock Lost Detection</i> subsection after <a href="#">Section 3.7.13.1</a> .....	92
• Changed OSCCLK cycles to INTOSC1 cycles in <a href="#">Section 3.9.3</a> .....	97
• Changed OSCCLK cycles to INTOSC1 cycles in <a href="#">Section 3.9.4</a> .....	97
• Added last sentence in <a href="#">Section 3.12</a> .....	105
• Added <a href="#">Section 3.12.1</a> .....	105
• Changed second paragraph in <a href="#">Section 11.2.3.2</a> .....	1008
• Added <a href="#">Section 11.13.7</a> .....	1039
• Added Note before Immediate Load Mode in <a href="#">Section 13.5.3</a> .....	1270
• Changed <a href="#">Figure 13-22</a> .....	1275
• Added <a href="#">Figure 13-83</a> .....	1344
• Changed Step 2 in <a href="#">Section 13.15.1.5.4</a> .....	1352
• Added <a href="#">Section 15.9</a> .....	1582
• Added last sentence to first paragraph in <a href="#">Section 16.1.3.4</a> .....	1634
• Added <a href="#">Figure 16-2</a> .....	1634
• Changed <a href="#">Figure 17-18</a> .....	1751
• Changed <a href="#">Figure 18-10</a> .....	1799

This page intentionally left blank.

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2024, Texas Instruments Incorporated